LEVEL II

2

INTEGRATED SCIENCES CORPORATION

Santa Monica, California

BEST

AVAILABLE

COPY

2

POST DETECTION TARGET STATE ESTIMATION USING HEURISTIC INFORMATION PROCESSING - A PRELIMINARY INVESTIGATION

Report No. 260-1

N00123-77-C-1260

By:

George J. Rebane,
Walter C. Gish,
Michael D. Schechterman
Gary W. Irving

SC-260-1

Prepared for:

Charles Irwin
Code 1226
Pacific Missile Test Center
Point Mugu, California 93042

September 1977

ABSTRACT

This report discusses methods of using heuristically derived representations of uncertainty to improve the performance of military command and control systems. Two algorithms are derived to permit system operators to define and evolve non-parametric probability density functions. These algorithms have been instrumented on an interactive computer-graphics system. This preliminary effort is posed as an initial step of a more extended research and development program.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

This report documents the results of a preliminary investigation of new post-detection target processing techniques applicable to command and control functions of swimmer defense systems (SDS). New interactive algorithms and computational techniques that come under the overall heading of heuristic information processing (HIP) were developed during the investigation. The algorithms developed can be used for estimating and predicting target location. They will ultimately comprise a technique which allows the system operator to generate his own target motion analysis solution or modify/edit the machine-derived solution. The output of these algorithms are generalized or non-parametric probability density functions (NPPDF's) which represent the target's location uncertainty. These PDF's are machine-usable and would serve as inputs to the succeeding phases of SDS functions such as the optimal allocation of sensor resources or computing a countermeasure tactic for the neutralization subsystem.

The remainder of Section 1 discusses some background material relevant to the investigation and formalizes the statement of the research problem. Section 2 presents a quick overview of the potential of heuristic information processing applied to swimmer defense and other command and control systems. In Section 3 we describe the interactive NPPDF algorithms instrumented for this project. The description of the interactive software is found in Section 4. Sections 5 and 6 discuss the results of a cursory engineering evaluation of these algorithms and includes some conclusions and recommendations for further study. The Appendices contain the program listings and technical materials which augment the presentation of the algorithms developed. Appendix D contains a summary description of an important experiment in heuristic information processing performed at ISC as part of the S2705 program to develop a defense against combat swimmers.

1.1 BACKGROUND

It is generally understood that the search and detection phases of the SDS mission are intrinsically probabilistic in nature. However, it is important to also understand that the post-detection phase of swimmer neutralization must use a probabilistic description of the problem. This is due to the fact that the target (threat) is neither perceived by the system as a deterministic point whose future actions are accurately known, nor can it be countered by elements of a neutralization subsystem whose performance is completely reliable.

The first post-detection/classification task to confront the system is to obtain a usable estimate of target parameters which will (1) confirm or establish the contact classification and (2) can be input to the decision and control algorithms which will, in turn, allow optimal deployment of the neutralization subsystem to counter the threat. The target state estimate (TSE) will contain such elements as present location and velocity, probable objective area, and the most likely path to be taken. In addition the TSE should provide measures of uncertainty related to each of these parameters in a form usable by both the operator and computer for deciding on the next course of action.

Obtaining the most accurate TSE requires (a) automatic-sensor-supplied inputs and (b) operator-supplied estimates of the situation which are based on a priori data and the operator's view of the current situation. In short, TSE generation requires the combination of data inputs which are inherently incompatible in their dimensions and formats. The correlation of this data by automatic means is far beyond the capability of foreseeable advances in the field or artificial intelligence. However a significant body of research is being done which indicates that the operator can contribute to the solution of the problem with the aid of an appropriate man-machine interface (MMI) design.

The nature of the resulting TSE obtained through the application of this emerging technology will most likely be a non-parametric representation of a dynamic probability density function (PDF) whose future location and shape can be generated using various types of data supplied by the operator as the situation develops. Figure 1 shows a typical display containing both the original sensor input data from a field of fixed sonobuoys, and a possible modification of a target's PDF. Also shown is an anticipated threat trajectory along with the expected terminal PDF at some future time as generated by the operator.

The Neutralization Subsystem (NS) of a modern SDS will most likely contain both preset and terminally guided deterrents/neutralization means which are deployed from a central Command and Control Center (CCC). A preset neutralization device (ND) is defined as hardware which, once launched/deployed, is no longer under CCC control. A terminally guided ND may, however, be controlled from the CCC after it is launched/deployed. In cases when the terminally guided ND is a weapon such as an anti-swimmer wireguided torpedo or a radio controlled boat dropping explosive charges, effective terminal control is required for successful system operation.

In Figure 2 we see a potential application of a heuristically-derived target PDF. This PDF represents the uncertainty in a target's locations as a function of time. The problem begins with an initial PDF at the left hand of the figure and ends with a terminating PDF at the right hand of the figure. The specific problem is to determine the optimum launch time of a neutralization device which is located as indicated in the figure. If the neutralization subsystem employs a straight-running, acoustic homing weapon which has a proximity type fuse, then the kill probability is proportional to the ratio of the shaded area to the total area of the PDF as indicated in the figure. It is seen that if the neutralization subsystem is constrained to remain at its current location, then the optimal deployment procedure is to wait until time T_3 and then lauch the weapon to achieve the highest kill probability. If, however, the neutralization subsystem can be moved during the countermeasure operation, then an even higher kill probability could be

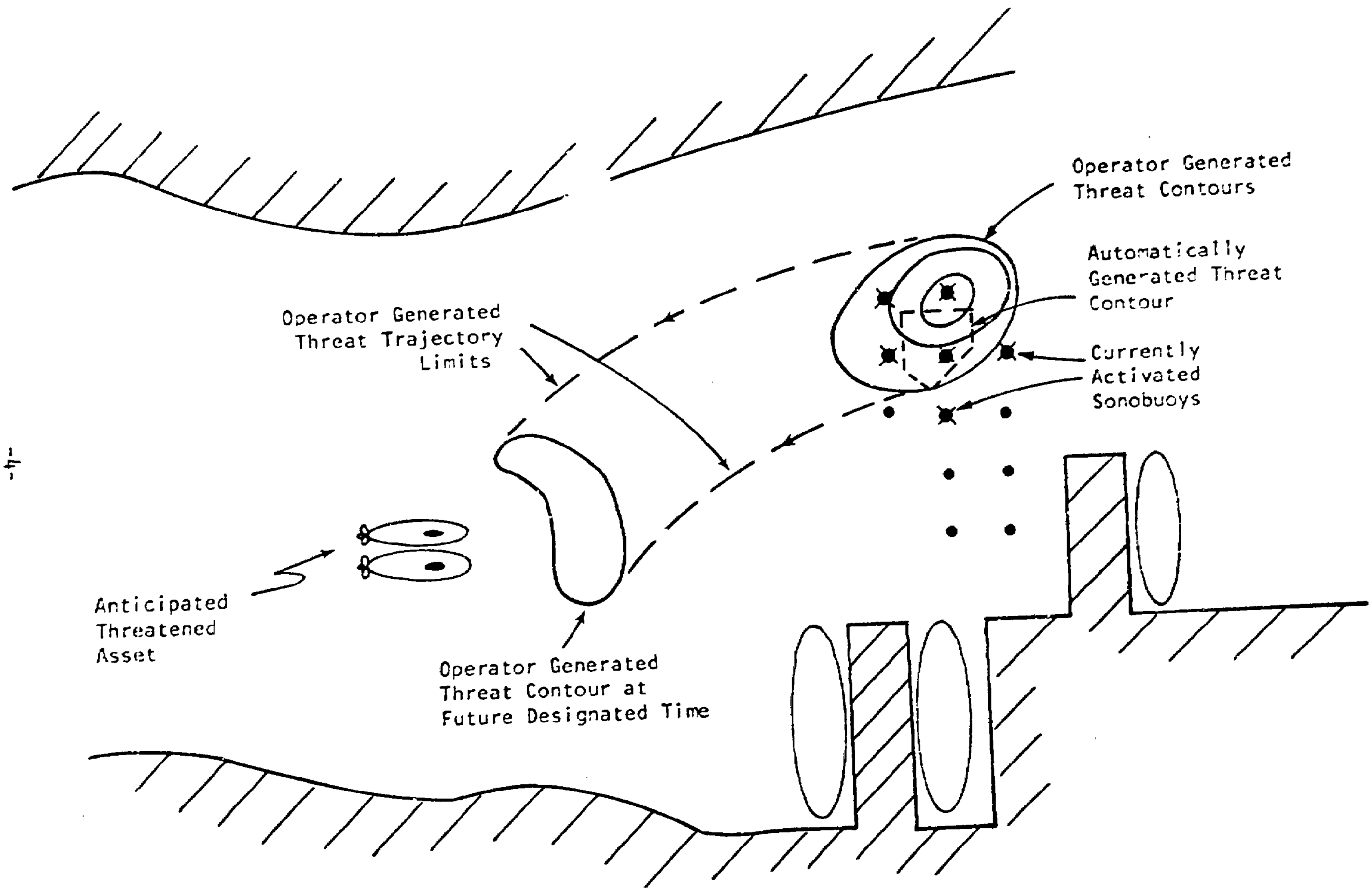


Figure 1. Possible Concept for Estimating and Predicting Threat State in a Harbor Defense Scenario.

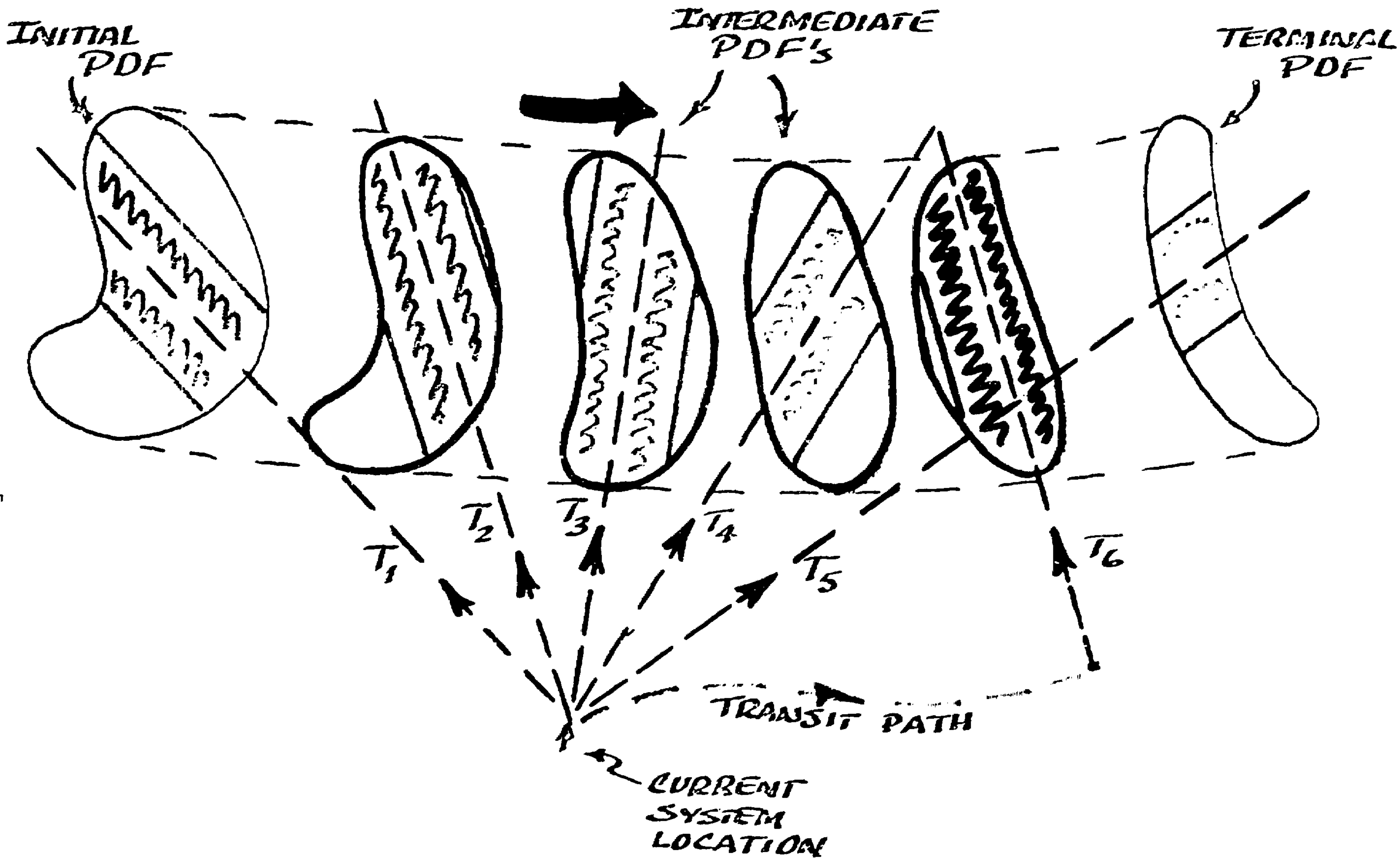


Figure 2. Optimal Search of Dynamic Non-Parametric Target Distributions.

achieved if the system is displaced as shown and the weapon launch is performed at T_6.

1.2 STATEMENT OF THE RESEARCH PROBLEM

The purpose of this work was to develop interactive methods for allowing a system operator to input data that permits the computer to generate a dynamic sequence of non-parametric probability density functions representing the target's motion. These NPPDF's would incorporate the "best information" provided objectively by the machine and subjectively by the system operator.

The data input by the SDS operator are heuristic representations of current target state uncertainty and the probable future evolution of the uncertain target states. The algorithms developed should allow the human to deal in a familiar tactical space through the use of an interactive graphics device. The graphical representations of uncertainty in target location and motion should be translatable into the probability density function format. In general these PDF's will be non-parametric, nevertheless the data thus derived should be machine usable for subsequent SDS functions such as sensor reallocation and neutralization subsystem control optimization.

2.0 SDS APPLICATION OF HEURISTIC INFORMATION PROCESSING

This section provides more detailed background information relevant to the specific tasks performed in this study than was given in Section 1. Much of the work which has contributed to development of this background information has been done at ISC in relation to a broader area of combat systems research than that performed for the SDS.

The following subsections introduce the primary differences between systems which are restricted to using parametric PDF's and potential future systems having the capability to incorporate NPPDF's in the decision making process. Examples in non-SDS areas are included to illustrate the wider use of this technology. We conclude this section by describing in non-mathematical terms how NPPDF's are generated and caused to evolve or move in the tactical time frame,

2.1 TACTICAL DECISION MAKING WITH PARAMETRIC PDF's

The use of deterministic algorithms to process tactically relevant data often results in unacceptable system performance. Recognition of this fact has caused the implementation of various forms of probabilistic models in many fleet systems. The algorithms used to represent uncertain knowledge require statistical assumptions to be made concerning the nature of the observed data and the underlying process describing the phenomenon under observation. These assumptions are seldom realistic.

The processed output of these algorithms represent the uncertainty in the system solution as a very regularly shaped parameterized distribution, usually multi-variate Gaussian. Experienced system operators, of course, realize that these parameterized distributions do not truly represent the total knowledge available about the problem. In other words, the human operator knows that the real world is not exactly represented by a combination of overlapping mounds of circular or elliptical shapes that have to interact with other distributions perhaps represented by rectangular boxes and circles of the so called "cookie cutter" family. Even though this simplistic representation of uncertainty serves to remind the operator that he is dealing with probabilistic instead of deterministic data, the operator is still forced to perform a parallel sequence of mental computations to correct the idealized distributions in the system.* This mental picture is a complex formulation of uncertainty which can no longer be accurately represented parametrically. Also it is not feasible to generate the theoretically obtainable composite PDF using automatic means because often the operator would modify the resulting nonparametric PDF with knowledge that does not reside within the system in machine-usable form. Furthermore, even if we begin with parameterized representations of tactical uncertainty and wish to proceed on some optimal course of action, we find that we are quickly forced to deal with non-parametric or more generalized representations of probability density functions.

Therefore the resulting modes of operation of these interactive tactical systems require a large measure of operator skill which must be obtained through (a) long on-the-job experience in developing the modified representations of tactical knowledge and (b) mentally performing a parallel processing of this knowledge. An analog of this process may be called application of modern "Kentucky Windage" techniques. These procedures using parametric PDF representations are inherently inefficient. Nevertheless they do provide a measurable performance increment over systems which do not account for the unreliability of either the input data or the intermediate system solutions.

2.2 THE NON-PARAMETRIC PROBABILITY DISTRIBUTIONS FUNCTION (NPPDF)

NPPDF's in a tactical system may be generated by the machine automatically, by the system operator, or by the system operator modifying machine generated PDF's. The most likely method by which NPPDF's will be generated during operational situations is the operator using a graphic input device supplied as part of the interactive tactical system.

---

*Also see Section 2.2.1 below.

We show in Figure 3 an example of how an operator may furnish information about the location of an object through the generation of a NPPDF. The figure shows a bi-modal NPPDF comprised of five contours. The outer contour designated C1.0 represents a region in the horizon place in which the system operator believes that the target is contained with a high degree of certainty. The next two nested contours labeled C.9 represent two regions of space where he believes the target may be found with probability of .9. The final two nested closed contours labeled C.5 represent regions of the space where the system operator believes the target may be found with probability 0.5. These contours termed Continuous Subjective Functions (CSF) or Sketch Models may be generated by the system operator using both quantitative and qualitative information about an object (target) whose location is not precisely known.

CSF's have already demonstrated their utility in the characterization of parametric probability density functions. This research performed at ISC* demonstrates that the human operator is capable of using (subjective) knowledge other than the objective knowledge presented by a set of observed sample points to describe the parametric distribution from which the sample was drawn.

The extension of the intuitively appealing conclusion of this work is depicted in Figure 4. Here we see three curves schematically representing the quality of a solution to a hypothetical tactical problem. The quality of the solution is plotted as a function of the amount of data available to the total system - man and machine. Three system operational modes are depicted. In the first mode the machine in operating autmoatically with no operator assistance. Here the machine requires a certain minimum amount of objective data prior to being able to generate any type of usable solution. The second mode involves only the opeator. It is known, of course, that the human can begin to generate solutions on very little or no objective

*See Appendix D for a more detailed discussion.

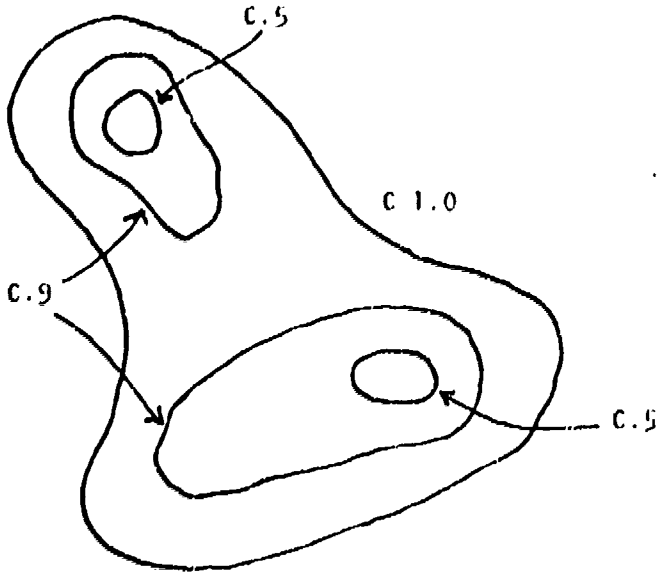


Figure 3. Operator Representation of a Non-Parametric Probability Density Function (NPDDF)

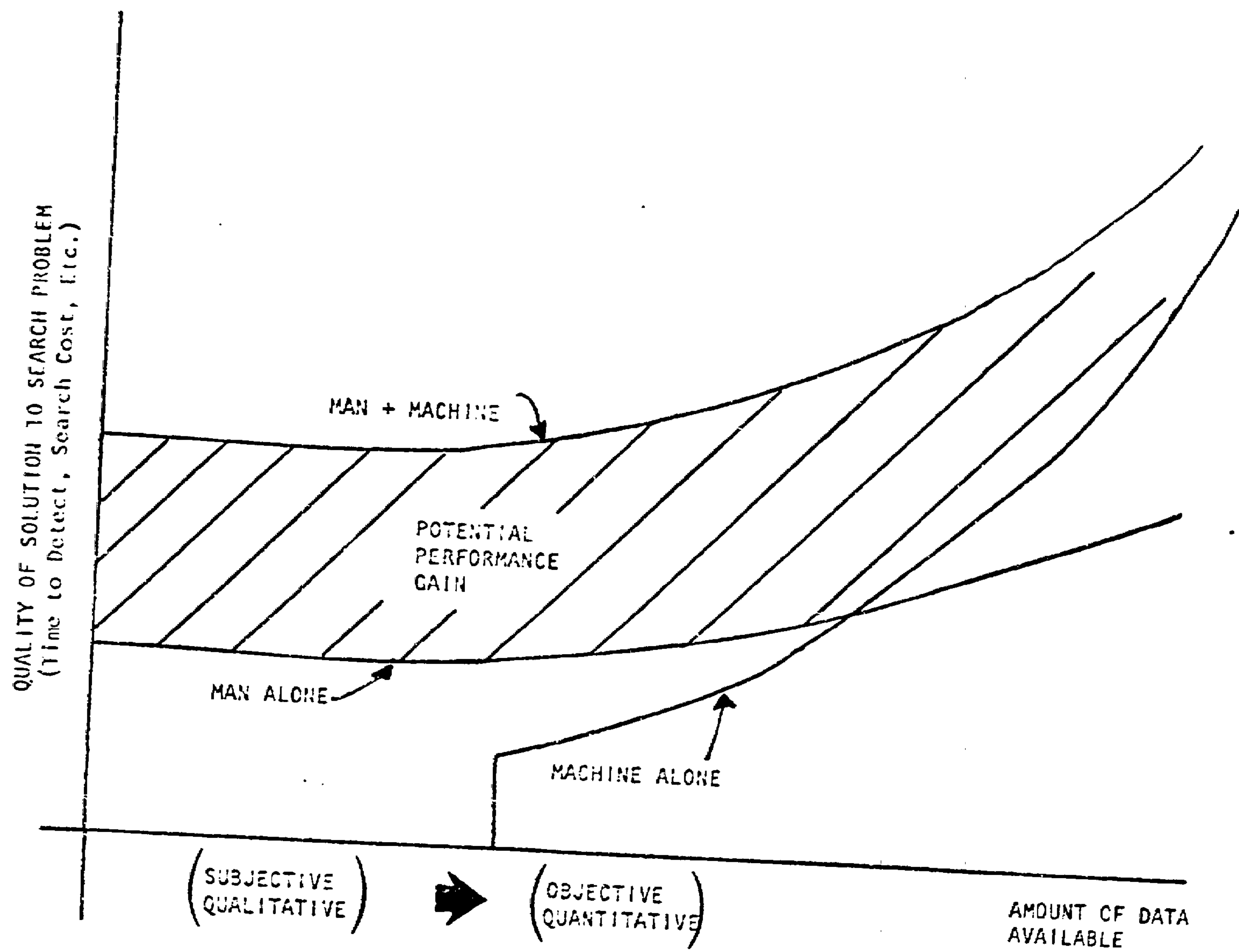


Figure 4. Potential Search Performance Improvement with use of Operator Generated NPPDF's

data. From sparse subjective data alone, usable solutions can often be obtained. (Often we term this reult also as "going off half cocked.") It is shown that for estimating parametric distributions from sampled points the machine operating alone could be outperformed by the human operator up to a certain sample size (amount of available data).* At that point a performance crossover occurred and the machine performed better.

We point out here that the above research was conducted using only parametric distributions to generate the operator displayed samples. It is easy to conceive of realistic tactical examples where non-parametric PDF's would occur in which a usable totally automatic solution would be impossible for the machine to obtain.

It is a fundamental hypothesis of the larger area of research which includes the current work that man and machine operating in concert can deliver a system performance gain which would make the quality of the man/machine solution asymptotically approach the machine-alone solution when an infinite amount of data was available to the total system. This is illustrated by the topmost curve in the figure.

In addition to the examples already discussed in preceding sections we examine a further example drawn from a mine countermeasures (MCM) mission. Figure 5 shows a horizon plane display of a minefield that is in the process of being cleared. Here the overall problem of the tactical MCM system is to develop time-optimal mine hunting policies that account for the mine sweeper kinematic constraints, mine hunting sonar unreliability, and navigational errors. The mission is to reduce the threat of the minefield to a predesignated level. The initial mine hunting tactic will most likely be developed using a uniform a priori distribution of the location of the mines in the minefields. Additional information will become available to the system operator after the operation in the minefield begins and several mines are detected, localized, and neutralized. This information would be used to update the mine hunting tactics in such a way as to most efficiently use the remaining resources.

*"Use of Interactive Graphics for Continuous Subjective Judgment," Doctoral Dissertation by Gary W. Irving, in progress.

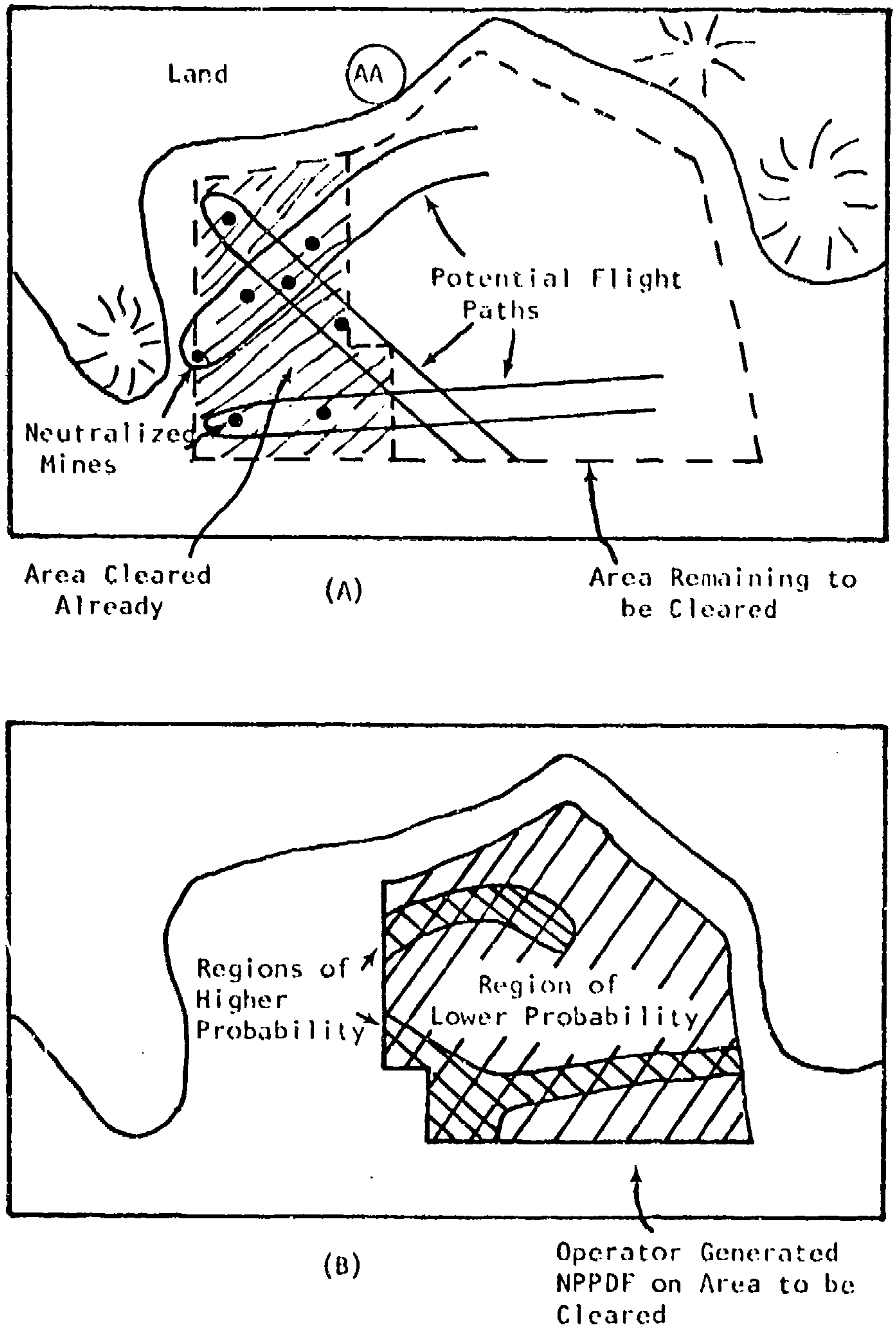


Figure 5. Operator Generated NPPDF for MCM Mission

In this example we have assumed that the system operator knows not only the location of the neutralized mines but also that the mines were laid by aircraft. This means that they lie in more or less linear patterns across the area to be cleared. If the minefield is in coastal waters where known anti-aircraft (AA) weapons and/or land features would preclude the mine laying airplanes flying certain paths or if other a priori knowledge is known from radar tracks of how airplanes crossed the minefield regions, then this information could be incorporated by the operator to generate NPPDF's designating the probable locations of the undiscovered mines. The second part of Figure 5 presents an operator generated NPPDF showing the probable locations of the undiscovered mines. With this machine-usable information the mine hunting system can now compute optimal search and scan trajectories that enable the required minefield threat level to be achieved in minimum time.

2.2.1 NPPDF Generation

The fact that today's systems are beginning to incorporate simple parametric representations of knowledge is due primarily to historical convention and the requirements of computational ease. In order to obtain implementable mathematical models of uncertainty in the past, the distributions had to have analytical simplicity. Today's systems are thus incorporating automated tactical decision models based on gaussian and "cookie-cutter" definitions of observation and process noise. And as shown by numerous simulation studies, these combat systems do perform better than their predecessors which were forced to "model the world deterministically." For many important command and control systems important questions still remain to be answered. These are:

1. What is the true performance increment of systems which parametrically model uncertainty, over systems which have no explicit analytical models of noise?

2. To what extent must these system-generated outputs be modified by man before they can be used to obtain an improvement in the overall solution to the tactical problem?

The combat system user often modifies the output or implements a neighboring solution, we feel, because he knows that the machine did not have the "true picture" of the situation. This behavior lends support to the hypothesis that in actual operational situations the representation of uncertain knowledge is almost always best represented by probability distributions of irregular shape (i.e., NPPDF's). Perhaps even a stronger impetus to study NPPDF's is drawn from the realization that even if the original PDF's at some point in the combat system were in truth parametric, they would become non-parametric as soon as we began operating with them. Therefore in obtaining any realistic optimal sequence of decisions we would be forced to treat NPPDF's in the process of solving the tactical problem.

We illustrate this in Figure 6 by examining the search for a target whose location is represented by a stationary bivariate normal distribution. For simplicity we will assume the sensor is perfectly reliable -- that is, if the target actually is in the snesor's observation area, then the sensor will report its presence with probability equal to 1.0. Since the observation area is "smaller" than the target's PDF, a sequence of observations from a moving sensor will generally be required. The figure illustrates the evolution from a parametric PDF (at time t_0) to a clearly non-parametric PDF (at time t_3) as the sensor proceeds with the search. The PDF is represented by the 0.5 equal likelihood contour* which initially (t_0) is an ellipse and then evolves into a bigger ellipse with a slice cut out as the unsuccessful search continues.

It is clear that if at, say t_2, the searcher decided to pick a new sweep course, or commit an additional sensor to the search, then he should do that with regard to the current state of knowledge at t_2 as represented by the NPPDF. In the same vein any formalized optimal search algorithm must also have the ability to treat NPPDF's if it seeks to treat the operational problem realistically.

*The 0.5 contour represents the region of space under a PDF where the searched for target is actually located with probability .5. The uniqueness of the contour for any shape of PDF is obtained by requiring all P-contours to coincide with the PDF's equal likelihood contours that satisfy the preceding definition. (See Section 2.4)

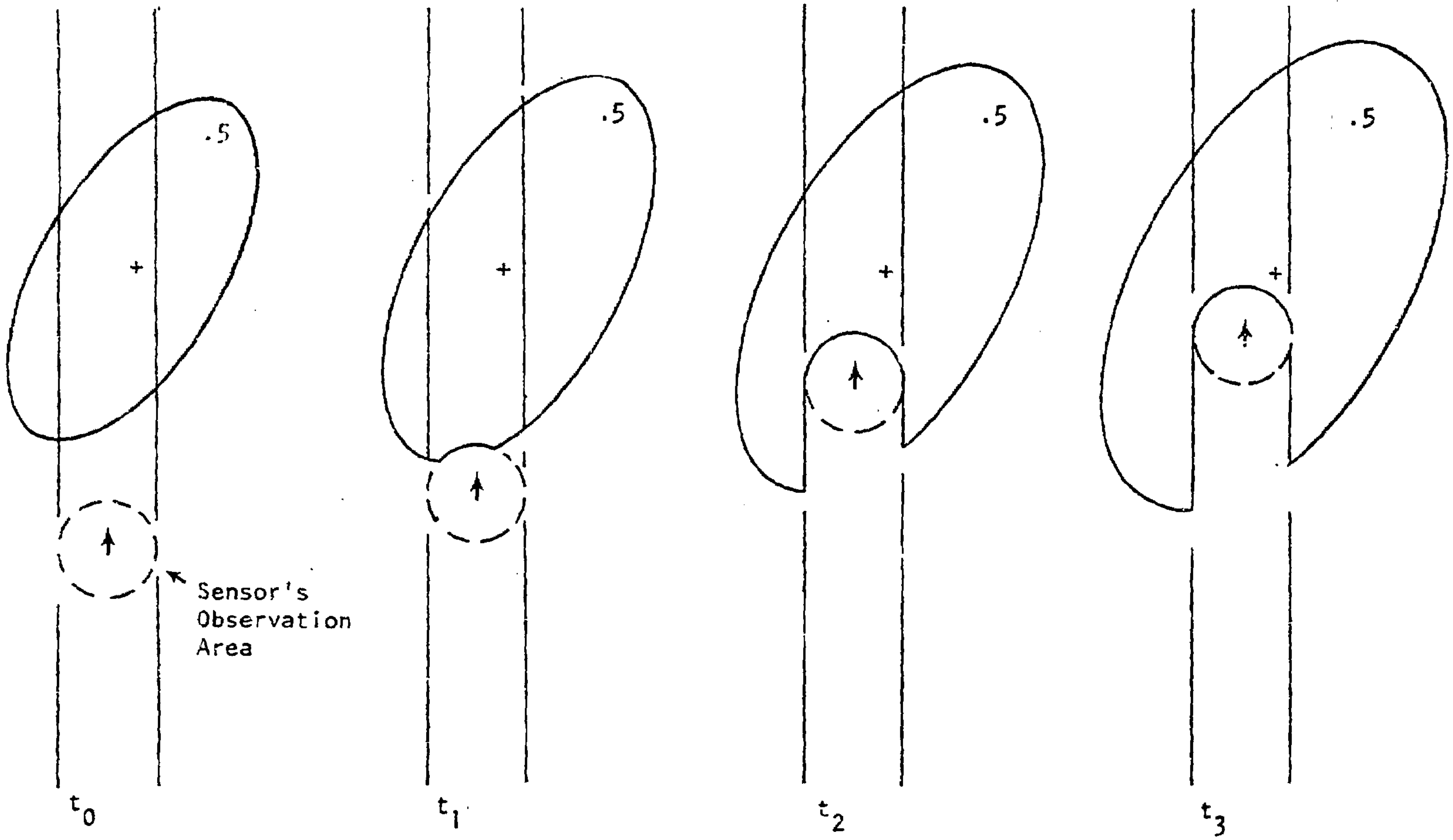


Figure 6. Search of a Stationary Bivariate Distribution with Perfect Sensor Having Circular Observation Area.

Another frequently occurring situation where NPPDF's arise is illustrated in Figure 7. Here we show the location of a target of interest as initially estimated by a Kalman filter. However, other a priori knowledge that was not processed by the filter algorithm (either because it was not capable of receiving or using the additional inputs) would cause the system user to modify the parametric representation of the target's location. He could do this mentally by identifying areas under the PDF which have a very low probability of containing the target. Were the system user able to supply the system with this data, the NPPDF shown in the right hand side of the figure would result. This distribution would now represent a more accurate statement of knowledge and, if properly used, would theoretically contribute to better system performance. This possibility is discussed in the remainder of Section 2.

In the real world of tactical systems the situation is somewhat different from that implied in the preceding paragraph. The system user's problem solving sequence currently employs a less efficient routine that is forced to rely on a cumbersome synergism between man and the machine. This situation is illustrated in Figure 8. Part A of the figure indicates a typical information flow that has a combat system processing measured environmental input and computing a solution. This solution may be used by the next system stage (either another problem solving system or a response system) as it is; or, as is usually the case, it is displayed to an operator who modifies the solution and passes it on as indicated by the switch position in the figure. In this way the overall system obtains the benefit of a more accurate representation of knowledge, although the likely result is that the overall solution suffers. This is so because the operator is forced to intervene in both the input and processing functions of the machine by modifying the system's solution. If this is done in a multi-stage process as shown in Figure 8(B), then a significant difference between the hands-on and hands-off solutions could occur with neither being necessarily the best solution possible. By "best solution" we mean one where the operator supplies inputs that incorporate information from diverse sources (not available to even the most visionary systems of the foreseeable future), and the machine performs the data processing functions to generate optimal solutions.

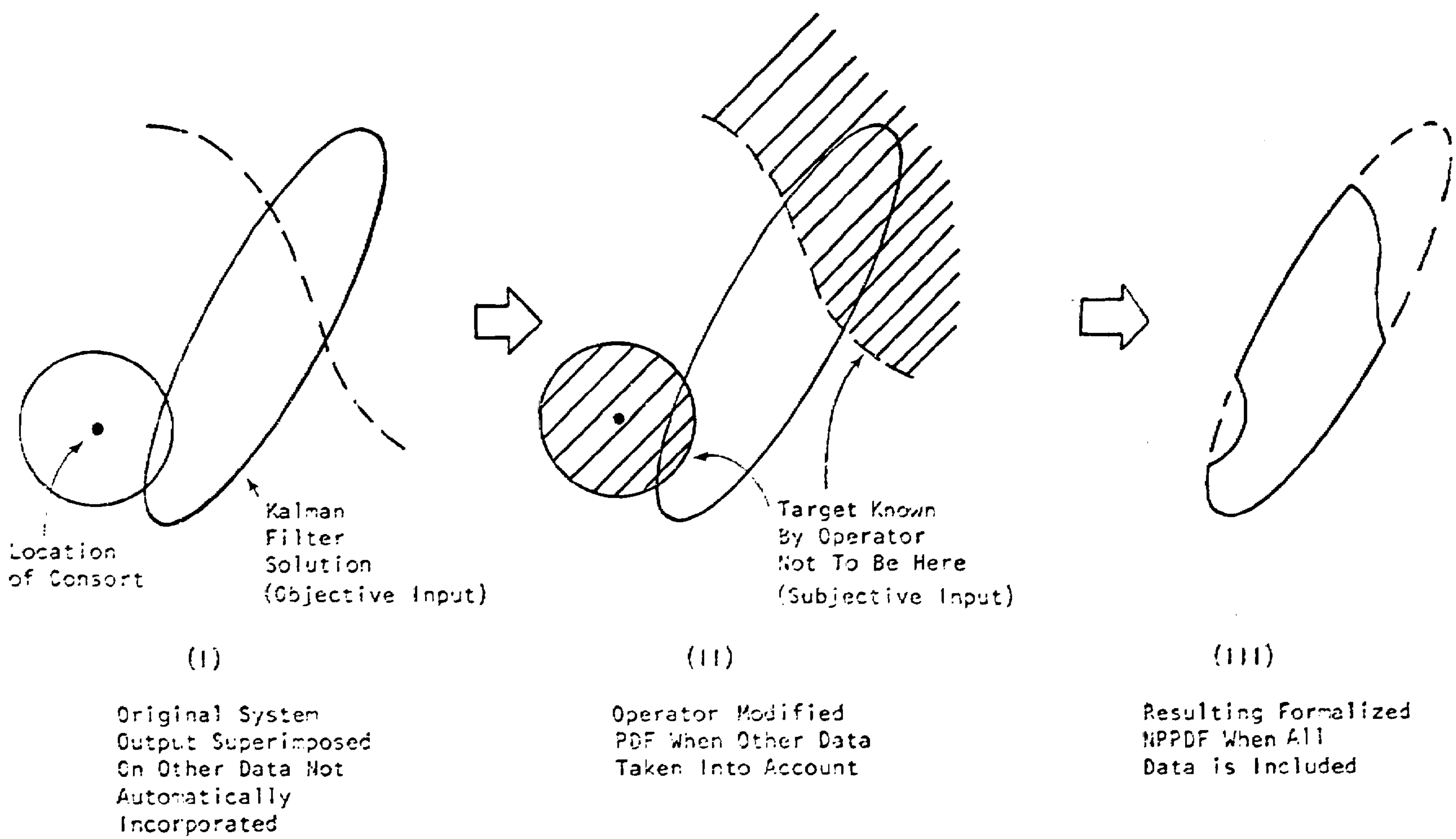
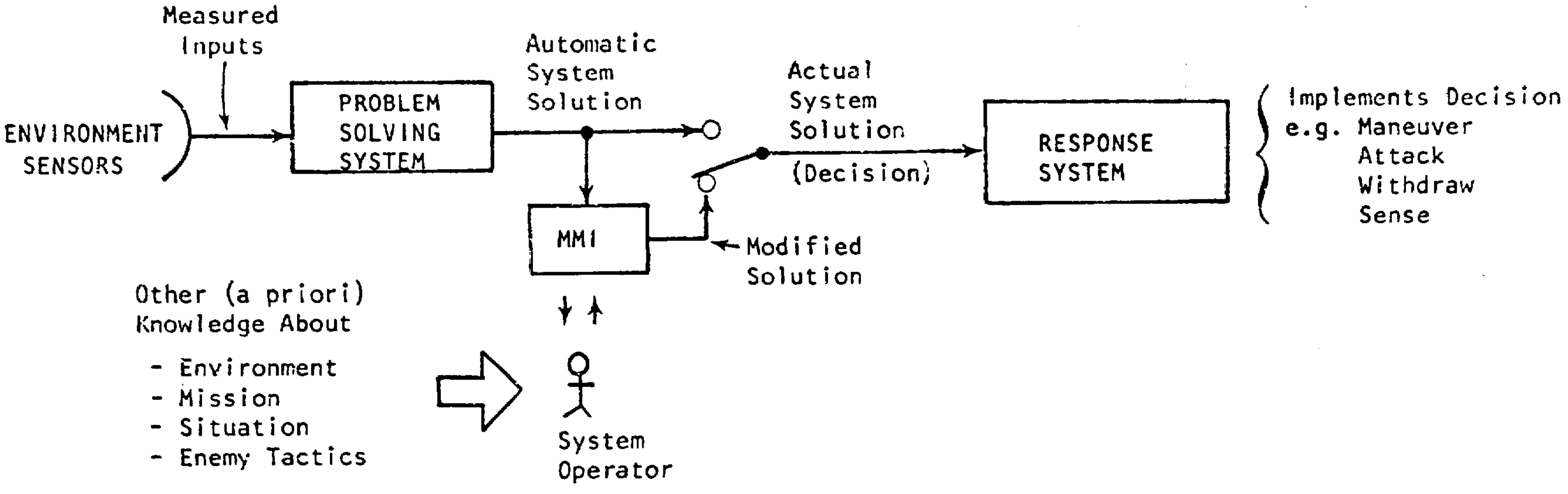
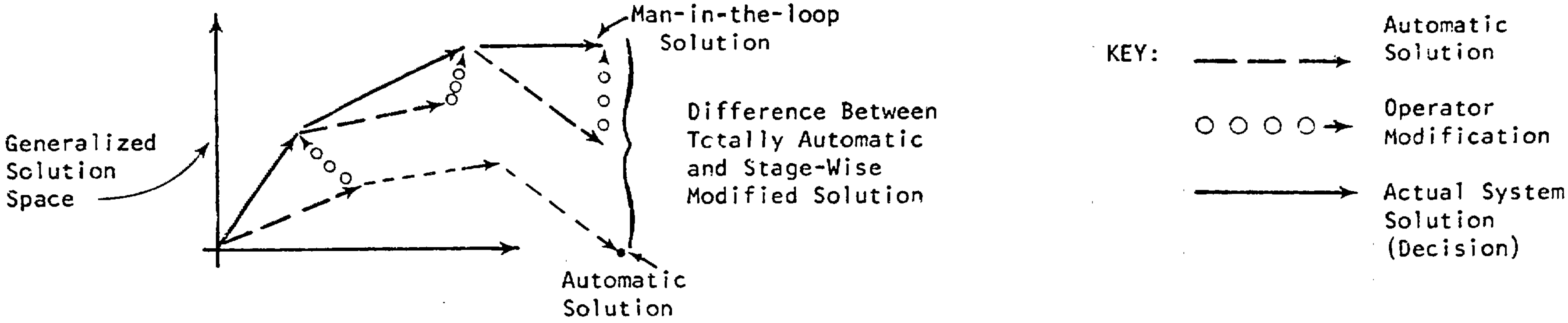


Figure 7. An Example of a NPPDF Generated by Using Data Not Available to or Processed by Current System Algorithms



(A) TYPICAL INFORMATION FLOW



(B) SCHEMATIC REPRESENTATION OF

GENERALIZED THREE STAGE SOLUTION

Figure 8. A Representation of the Generation of Tactical Problem Solutions in Current Systems.

2.2.2 NPPDF Dynamics

There are two ways to examine the future dynamic behavior of NPPDF's. One way is to define (a) an initial PDF at some time t_0 and a terminal PDF at some time t_1, and (b) a nominal path of how the PDF evolved from t_0 to t_1. An algorithm using statistical sampling of these terminal NPPDF's can then be used to generate an intermediate PDF as shown in Figure 9. In such fashion a number of intermediate PDF's could be generated. Another algorithm described below can be used now to continuously "evolve" each contour segment between the defined and/or sampled intermediate PDF's.

A second method of predicting the future behavior of NPPDF's is to define an original PDF specifying, say, the positional information of a target and define a derivative distribution which characterizes the uncertainty in the knowledge of the target's velocity. This is shown in Figure 10. A new terminal PDF can be generated at some time in the future by sampling both the locational and the derivative distributions and applying the derivative distribution value to the locational sample. The smooth transition from the original supplied NPPDF and the terminal distribution would be obtained through a specialized algorithm described below.

The contour interpolation algorithm summarized in Figure 11 permits the smooth interpolation of an arbitrary initial contour to a terminal contour in such a way the the evolution of the intermediate contours is controlled by two designated constraint boundaries. The algorithm uses these input data to numerically compute a grid of internal points that define what may be called minimal polynomial designators with points on the terminal contour. These designators may be visualized as being akin to the Lagrangian solution of field lines or hydrodynamic flow streamlines. If a more complex evolution is required between initial and terminal contours, then an intermediate contour showing the more complex solution can be defined. The interpolated contour now will proceed from the initial contour through the intermediate contour to the terminal contour. It is also possible to vary the transit speed in the contour interpolation algorithm. A graph of this is shown in the figure where the transit speed of the evolving contour could be a non-linear function of the along-boundary distance.

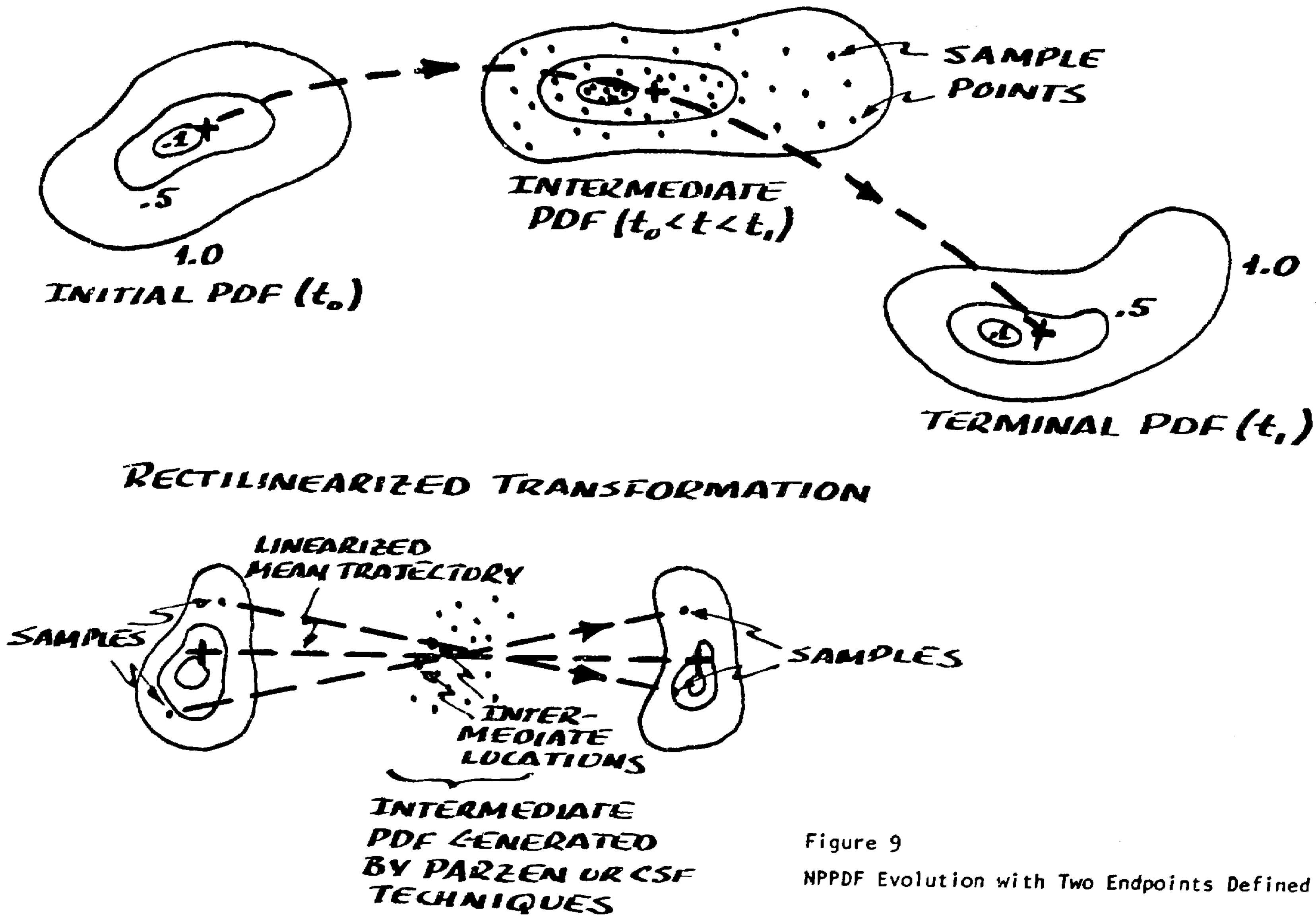


Figure 9

NPPDF Evolution with Two Endpoints Defined

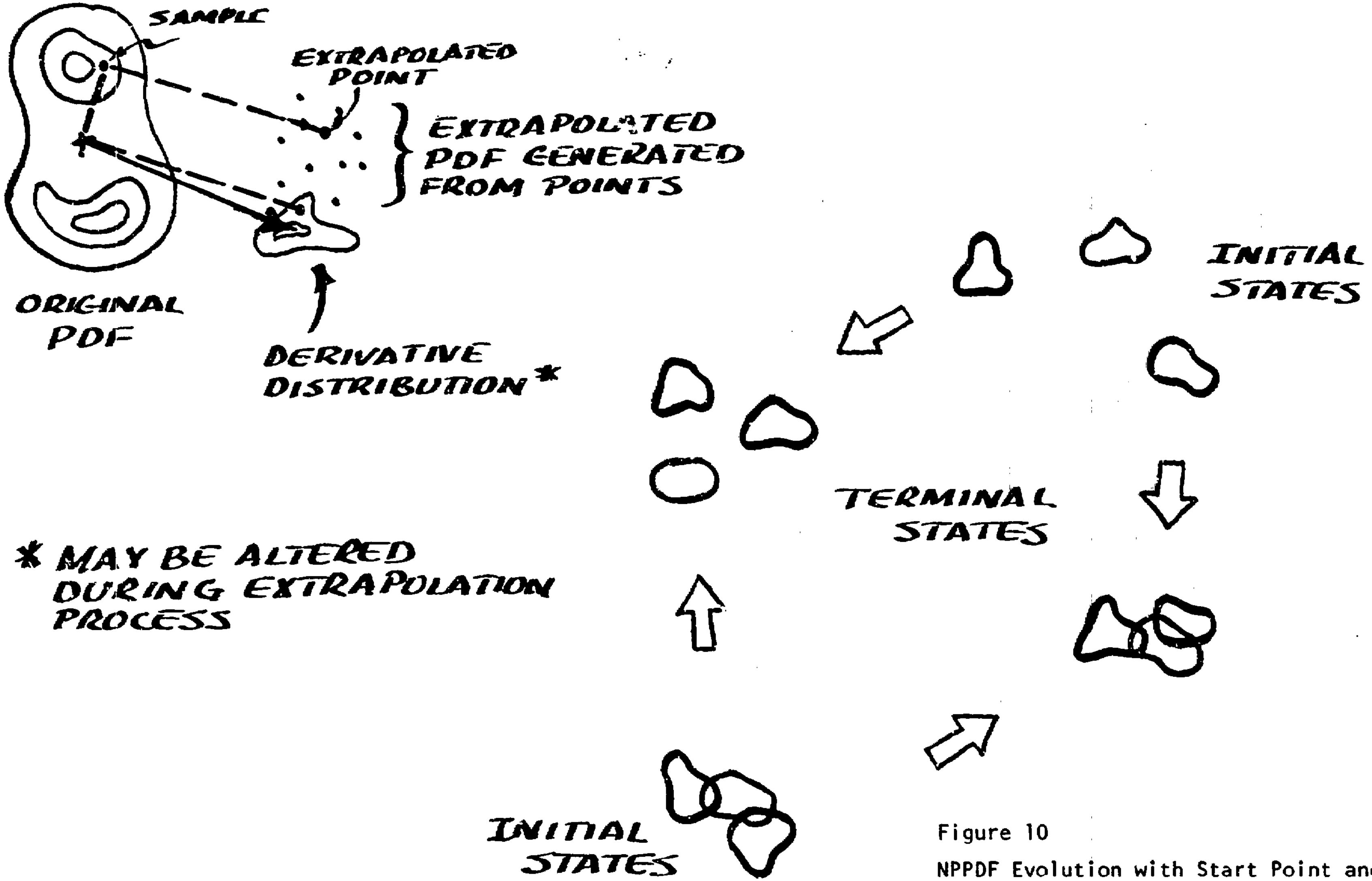


Figure 10

NPPDF Evolution with Start Point and Derivative Distributions Defined

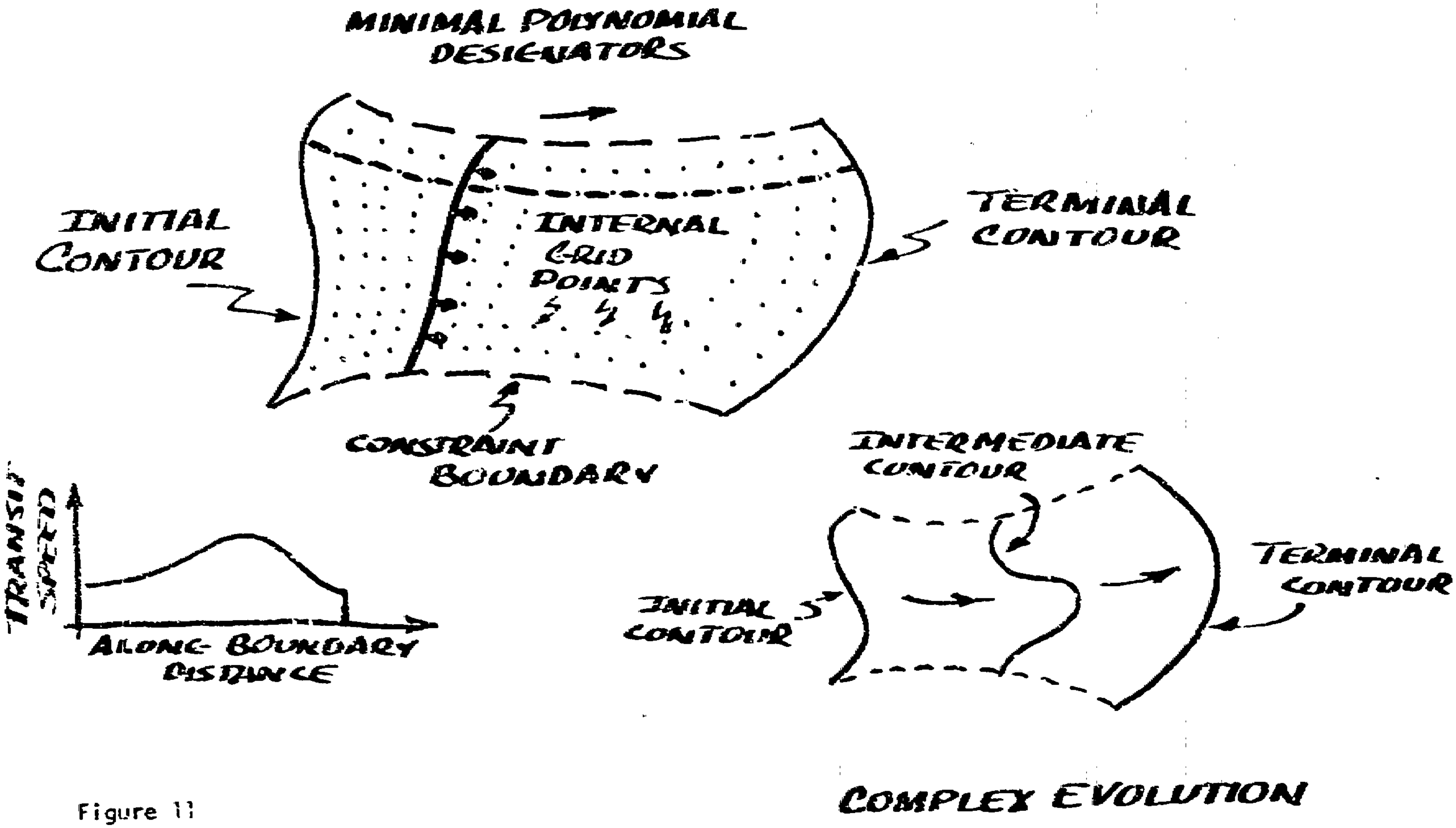


Figure 11

Contour Interpolation

The technique summarized in Figure 11 would be used to govern the dynamic behavior of contour segments of NPPDF's that were obtained through the processes summarized in Figures 9 and 10.

Before concluding this section we re-examine Figure 10. On the lower right hand side of the figure we see two combinations of three estimates of initial states evolving into two possible terminal combinations. This is a schematic representation of how subjectively supplied information may be examined with a tactical system having the capability to manipulate NPPDF's. The schematic shows that the definitions of the three initial states provided by various members of the commander's staff may have a large degree of concurrence as shown in the lower part of the figure. On the other hand, there may be wide variance as shown in the upper part of the figure. Each member could define not only the locational distribution but also his best estimate of what the derivative distribution is and allow the machine to evolve the future states of uncertainty as shown. It would then be possible to see that the individual initial states as they evolve could either remain together or disperse. Presentation of these results to the commanding officer (decision maker) would allow him to deduce the degree of concurrence or variance of the estimates of the situation existing among his staff.

3.0 DESCRIPTION OF INTERACTIVE NPPDF ALGORITHMS

Two different approaches to the interpolation of non-parametric probability distribution functions have been implemented. The first of these interpolates contours constrained by boundaries, and is described in Section 3.1. The second algorithm interpolates between individual samples of the initial and final distributions. This interpolation takes place in a linearized coordinated system determined by the operator input of the distributions mean. This algorithm is described in Section 3.2.

Note that both algorithms described below are combinations of

a) Simple geometric transformation, and
b) Linear interpolation;

and hence might be expanded to utilize more complex transformations or other interpolation schemes.

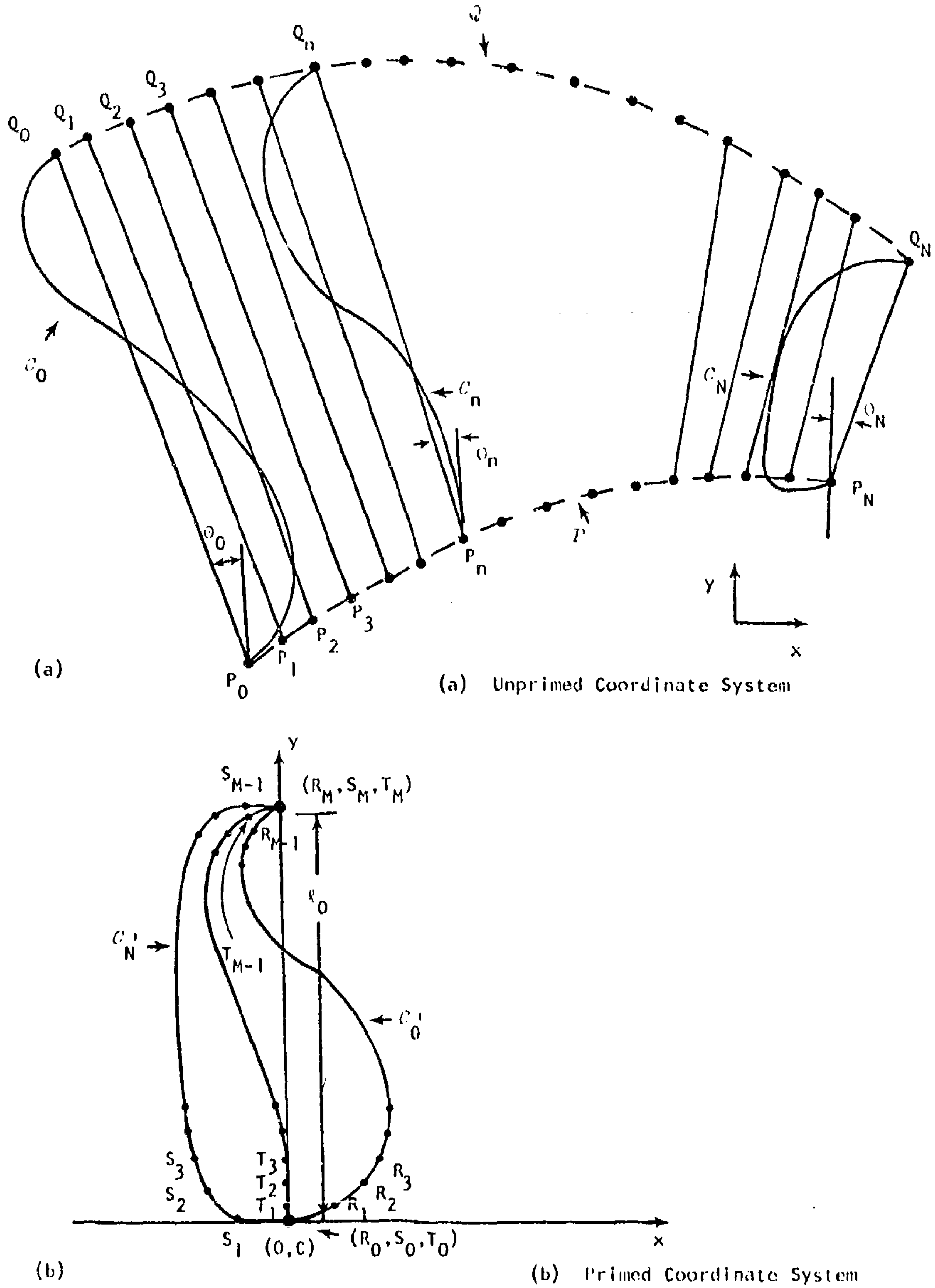
3.1 The Contour Interpolation Algorithm

The contour interpolation algorithm, shown in Figure 12, assumes initial and final contours (C_0 and C_N) and also transition boundaries (P and Q) describing the evolution of the endpoints of the contours are given as shown in Figure 12(a). (a). There are four basic steps to this algorithm:

1. The transition boundaries are divided into N equal length segments.

2. The initial and final contours are transformed to a new coordinate system indicated by primed variables. The initial contour is moved so that its bottom endpoint is at the origin, and rotated so that the other endpoint is along the Y-axis. The final contour is similarly displaced and rotated, but, in addition it is stretched or shrunk so that its chord length (endpoint to endpoint) agrees with that of the initial contour as shown in Figure 12(b). The end result is that the initial and final contours have been transformed so that their endpoints match.

3. An intermediate contour in the primed coordinate system is calculated by interpolating between the initial and final contours in the primed system.



(a) Unprimed Coordinate System

(b) Primed Coordinate System

Figure 12. Contour Interpolation Algorithm.

4. The final intermediate contour is a rotated and stretched/shrunk version of the primed intermediate contour displaced back to its appropriate endpoints in the unprimed coordinate system.

Again, referring to Figure 12(a) and (b), the equations used to perform these steps are:

1(a) Divide the transition boundaries into N equal segments specified by points $\{P_n\},\{Q_n\}$, $n = 0,N$.

1(b) Calculate ℓ_n, the distance from P_n to Q_n, and Θ_n, the angle of the line $\overline{P_n Q_n}$ from the vertical, for $n = 0$ to N

2(a) Calculate $\mathcal{C}_0'$. Move P_0 to the origin and then rotate $\mathcal{C}_0$ through Θ_0.

2(b) Calculate $\mathcal{C}_N'$. Move P_N to the origin, rotate $\mathcal{C}_N$ through Θ_N and multiply y coordinates by $\frac{\ell_0}{\ell_N}$.

2(c) Divide $\mathcal{C}_0'$ and $\mathcal{C}_N'$ into M equal length segments specified by points $\{R_m\}$, $\{S_m\}$ $m = 0,M$ respectively.

3 Calculate the points $\{T_m\}$ which specify $\mathcal{C}_n'$, a linear interpolation between $\mathcal{C}_0'$ and $\mathcal{C}_N'$.

$$\underline{T}_m = \frac{(N-n)}{N} \underline{R}_m + \frac{n}{N} \underline{S}_m$$

where the underline indicates the vector notation for the points.

4(a) Stretch/Shrink $\mathcal{C}_n'$

$$\underline{T}_m = \frac{\ell_n}{\ell_0} \underline{T}_m$$

4(b) Rotate through $-\Theta_n$ and displace back to P_n, the n^{th} point on the lower transition boundary.

3.2 The Sampling Algorithm

The Sampling Algorithm assumes that the isoprobability contours describing the initial and final distributions along with a description of the mean path are given. Again, there are four basic steps to this algorithm:

1. Sample the initial and final distributions.
2. Transform these samples to a primed coordinate system. In this case the primed coordinate system is one where the mean path has been transformed onto the x-axis.
3. Interpolate between these transformed samples to find the transformed intermediate samples.
4. Transform the intermediate samples back to the original coordinate system.

Now, referring to Figure 13, the equations used to implement this procedure are:

1(a) Assume initial contours $\{C_n\}$, and final contours $\{D_n\}$ with cumulative probabilities P_n describing regions $\{R_n\}$, $\{S_N\}$, for $n = 0,N$. P_0 must be 1.0, and P_n must be monotonically decreasing. Pick a total sample size M, and calculate the number of samples I_n required in each region R_n or S_n

$$I_n = \left[\left(P_n - P_{n-1}\right) M + 0.5\right]_{\text{truncation}} \qquad n = 1,\ N-1$$

$$I_N = \left[\left(P_n\right) M + 0.5\right]_{\text{truncation}}$$

Adjust I_0 to maintain a total of M samples

$$I_0 = M - \sum_{n=1}^{N} I_n$$

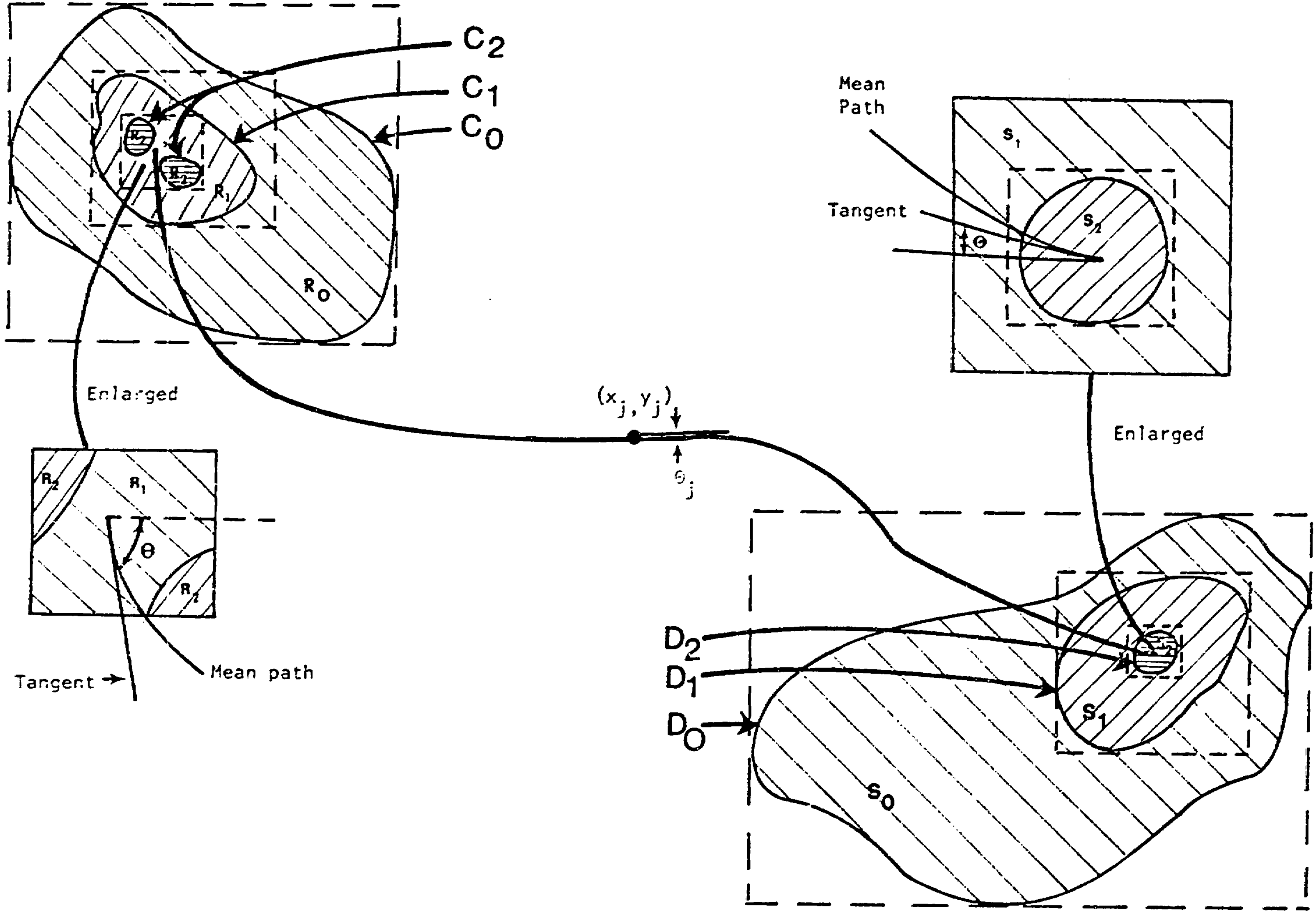


Figure 13. Sampling Algorithm (Unprimed System)

1(b) Circumscribe each contour $\{C_N\}$ and $\{D_N\}$ with a rectangle.

1(c) Begin with the initial distribution. Pick a random sample from the largest rectangle as follows: Let the rectangle be described by x_{min}, x_{max}, y_{min}, y_{max} then the sample coordinates are

$$x_{xample} = x_{min} + (x_{max} - x_{min})\ Z_1$$

$$y_{sample} = y_{min} + (y_{max} - y_{min})\ Z_2$$

where Z_1 and Z_2 are random numbers in the range [0,1].

Determine if the sample is in R_0, R_1, R_2 or outside all the contours. If it is in a region R_K and the quota I_K for that region has not been filled, save the sample point; if not, pick another sample and repeat. Once the outermost region R_0 has been filled, pick samples from the rectangle enscribing R_1. Then once R_1 has been filled use the rectangle enscribing R_2 and so on until all the quota I_n have been filled. Do the same for the final distribution.

2(a) Move the initial samples to the origin and rotate them through Θ_i (the angle between the mean path and the position x-axis direction), and call the rotated samples $\{U_k\}$, k = 1,M as shown in Figure 14.

2(b) Similarly, referring again to Figure 14, rotate the final samples through Θ_f, then move them out to $(\ell_p,0)$ where ℓ_p is the length of the mean path, and call them $\{V_k\}$, k=1,M.

3 Interpolate between $\{U_k\}$and $\{V_k\}$ according to path length to obtain intermediate samples $\{W_k\}$

$$W_k = \frac{\ell_p - \ell_j}{\ell_p}\ U_k + \frac{\ell_j}{\ell_p}\ V_k \qquad k = 1,M$$

where ℓ_j is the distance from the origin to point j on the mean path.

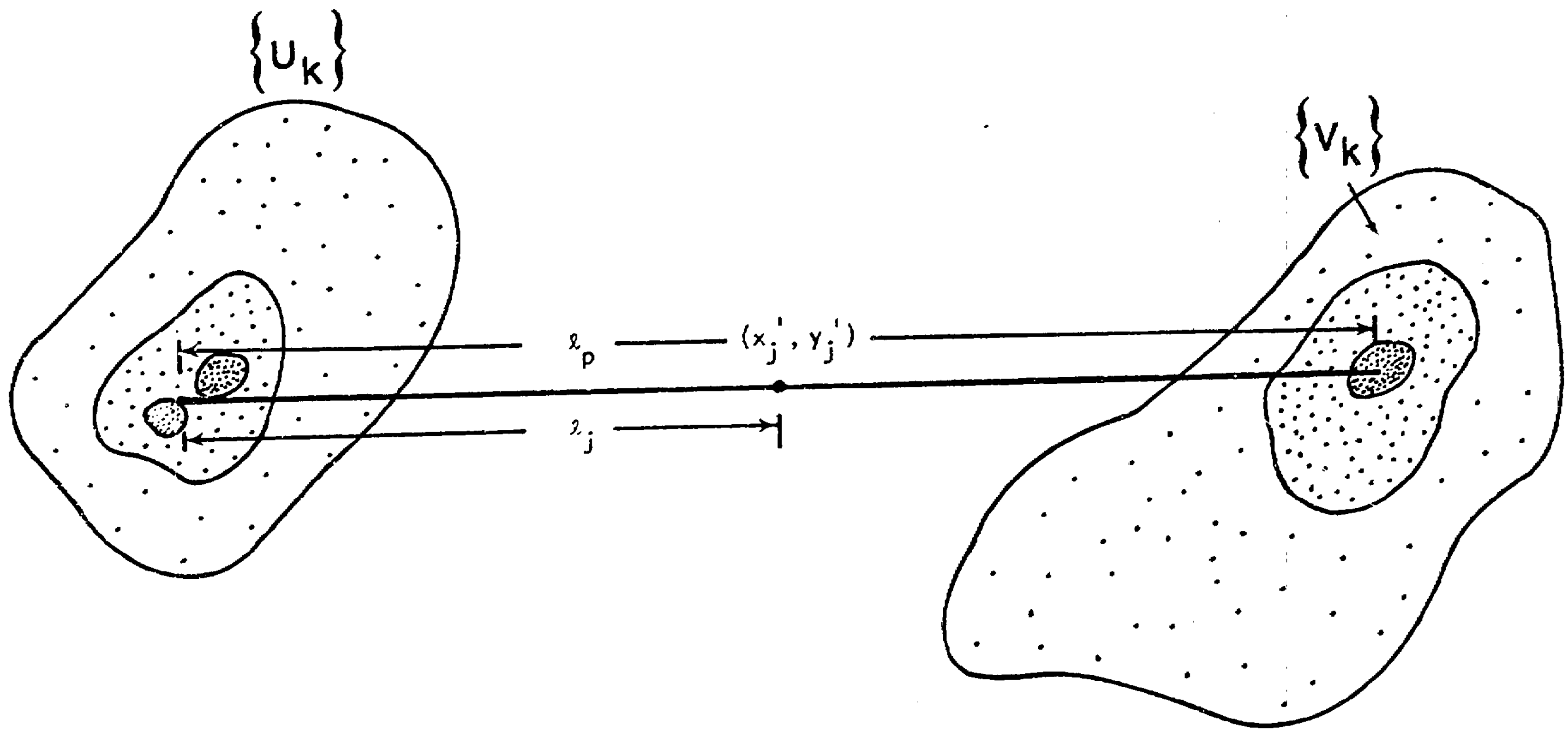


Figure 14. Sampling Algorithm - Prired Coordinate System

4 The intermediate samples are then rotated through θ_j, and displaced to (x_j, y_j).

The next stage of processing would be to draw isoprobability contours E_n, $n=0,N$ for this intermediate distribution. This requires the estimation of the actual intermediate probability density function from the set of samples $\{W_k\}$, $k=1,M$. Because we are concerned with complex distributions, simple estimators such as that proposed by Parzen are not sufficient. Fortunately, the variable kernel approach recently proposed by Breiman, Meisel, and Purcell (See Appendix C) seems appropriate. However, the results of our own testing do not replicate the qualitative behavior described by the authors even though the final accuracies are similar. Because of this, no NPPDF estimator has been incorporated in the sampling algorithm. Also, it must be noted that the algorithm as originally proposed requires approximately $10N^2$ exponential function calls. With 200 μsec required to generate an exponential on a V73 computer with floating-point hardware, a sample size of $N=400$ points would take 6 minutes of processing time. The development of an adequate contour algorithm therefore will be undertaken during a succeeding phase of this research.

4.0 SOFTWARE

Both the contour interpolation and the sampling algorithms have been programmed for a Varian V-73 minicomputer equipped with an Information Displays 4-color vector interactive graphics terminal, trackball, and function keyboard (See Figure 15). Both algorithms have the following characteristics:

- Highly interactive
- Use of color to enhance clarity of displayed information
- Operator entry of contours via trackball
- Program control via function keyboard.

The following two subsections describe the operation of the software. Actual listings of the programs are provided in Appendices A and B.

4.1 The Contour Interpolation Program

The operation of this program requires an operator to first enter initial and final contours, and then the upper and lower transition boundaries using the trackball associated with the display. Once these are drawn the program begains automatically and can be halted, continued, or restarted at will.

The specific steps are (Referring to Figures 16 and 17):

1. Initially, a cursor (two concentric circles) and the message "MOVE CURSOR" appear on the screen. The operator then uses the trackball to position the cursor to the starting point of the initial contour. When ready, the button at the lower right corner of the function keyboard labeled "MOVE/DRAW" is pressed (See Figure 17).

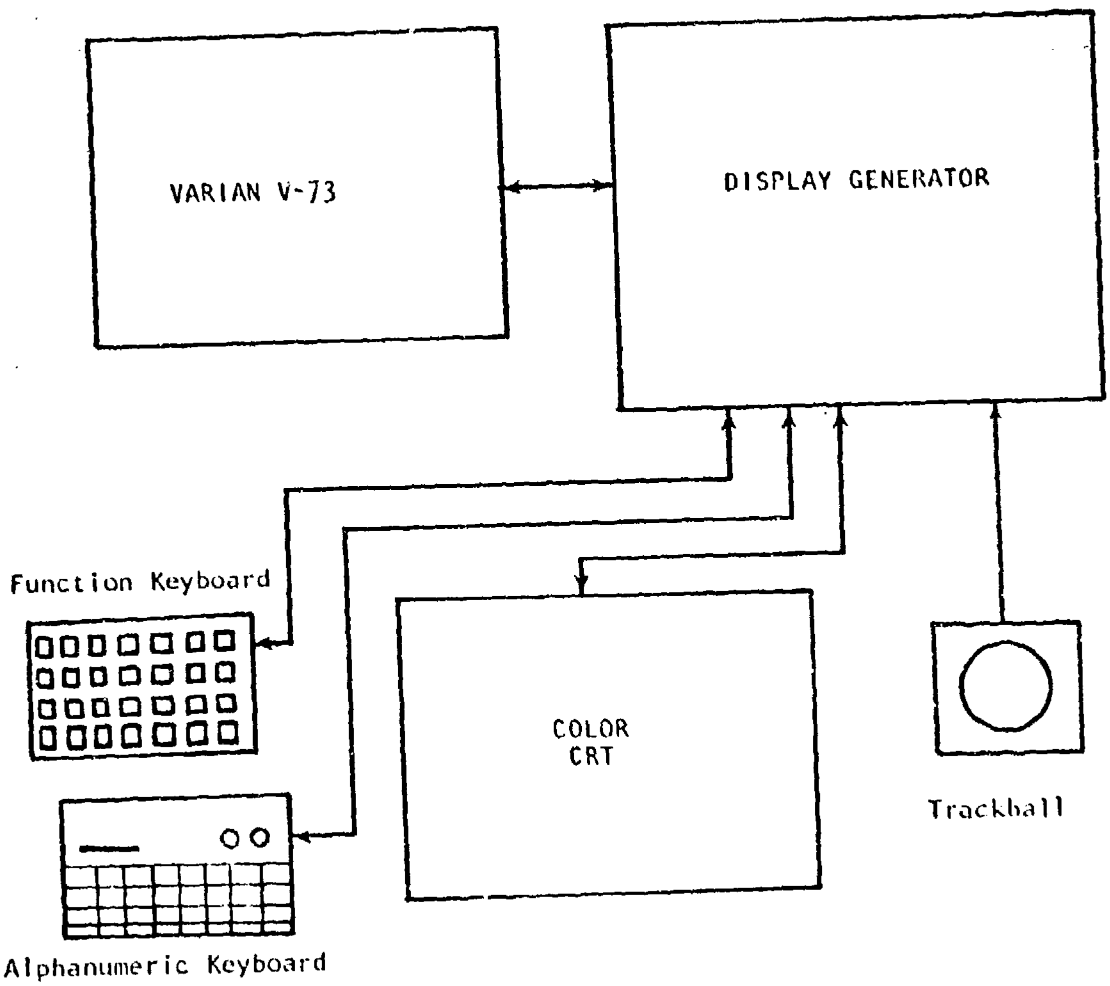


Figure 15. Computer System for NPPDF Interpolation Software

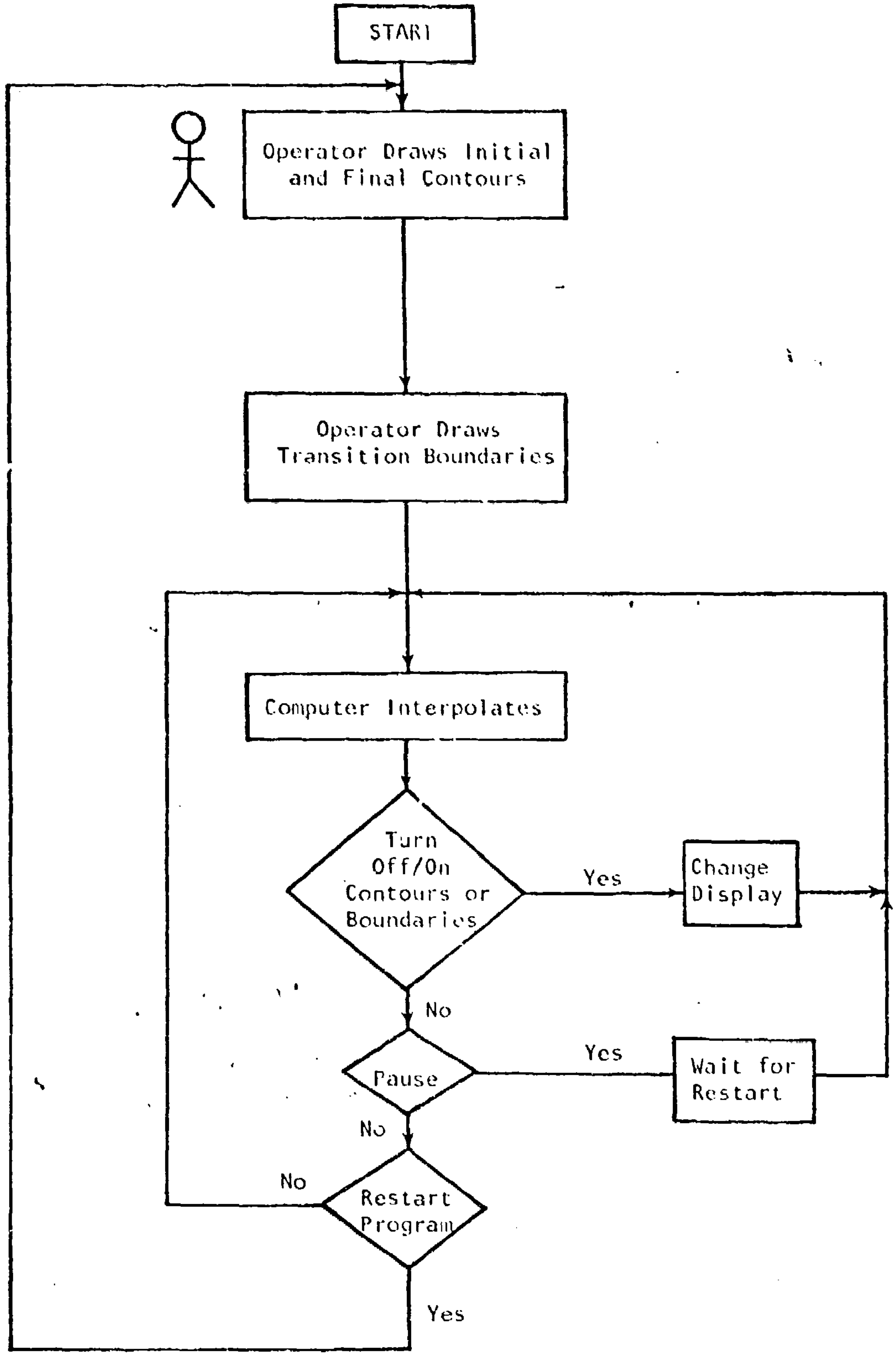


Figure 16. Operator Flowchart for Contour Interpolation Program

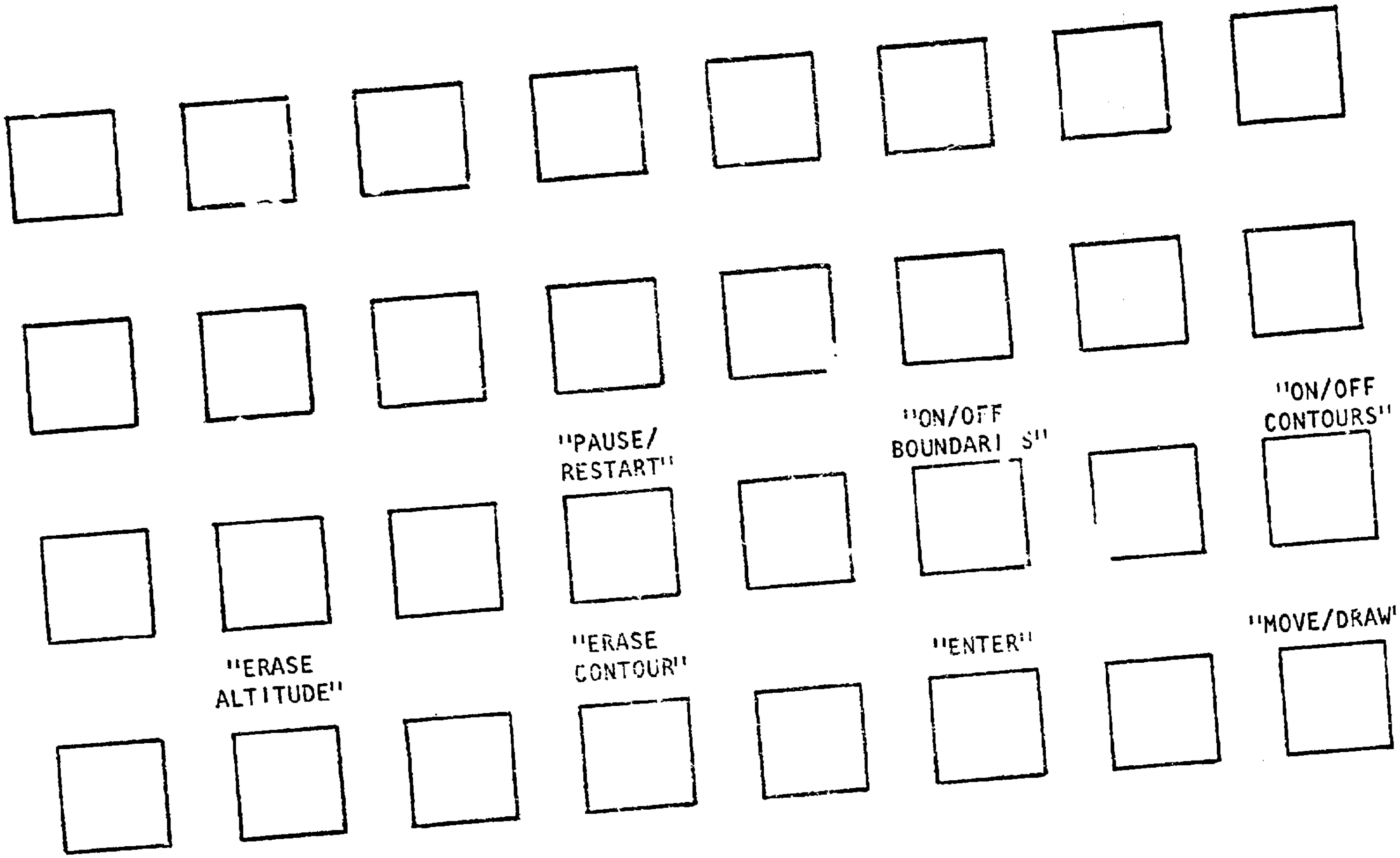


Figure 17. Function Keyboard Layout for the Contour Interpolation Program

2. The message "DRAW CONTOUR" is now displayed. The operator then uses the trackball to draw the initial contour (contours are colored red). Drawing continues until "MOVE/DRAW" is pressed again. This terminates the contour. Four keys are now lit. If the operator is satisfied with the contour drawn, he hits "ENTER" to enter this contour into the computer. If the operator wishes to redraw this contour, he presses "ERASE CONTOUR" or "ERASE ALTITUDE" and returns to Step 1.

3. When the initial contour has been entered, the message "MOVE CONTOUR" again appears. The operator then draws the final contour as in Steps 1 and 2. Before pressing "ENTER" to enter the final contour drawing, but after pressing "MOVE/DRAW" to terminate it, it is desirable to move the cursor to the left (beyond the initial contour) to give trackball "room" for the following step.

4. When the final contour has been entered, a cursor appears at the starting point of the initial contour. The operator then draws the upper transition boundary using the trackball. The transition boundaries are colored green. The drawing will automatically end when the cursor reaches the starting point of the final contour. The cursor should then be moved to the left again before pressing "ENTER" or either of the erase buttons.

5. The lower transition boundary is then drawn as in Step 4.

6. Once the lower transition boundary has been entered the interpolation begins automatically. Three keys are lit: "ON/OFF END CONTOURS," "ON/OFF BOUNDARIES" and "PAUSE (RESTART)." The The first two of these allow the operator to turn off the display of either the initial and final contours, or the transition boundaries. "PAUSE/RESTART" freezes (halts) the interpolation process; pressing it again restarts the interpolation where it left off.

   PRESSING ANY OTHER KEY CAUSES THE PROGRAM TO RETURN TO STEP 1.

Note that the interpolation uses smoothed contours and boundaries, so that the interpolated contours do not exactly correspond to the drawn ones and their transition boundaries. The interpolated contours are green.

4.2 The Sampling Program

The operation of the Sampling Program falls easily into 3 parts - input of the initial and final distribution, input of the mean path between the two distributions, and specifying the point on the mean path at which we wish an interpolated distribution. Figures 18 and 19 contain the operator flowchart and the function keyboard layout for this program. The individual steps within each of these parts are:

A. Specifying the initial and final distributions

1. Computer displays "INPUT INITIAL CONTOURS."

2. Computer queries "NO. OF CONTOURS" and the operator enters an integer N less than or equal to the maximum number of allowed contours (currently 5).

3. M = 1.

4. Computer queries "CUMULATIVE PROBABILITY FOR CONTOUR M" and the operator enters an integer N less than or equal to 1 or the last cumulative probability.

5. The computer displays "DRAW CONTOUR" and "MOVE CURSOR." The operator moves the cursor to the starting point of the contour and presses function keyboard button labeled "MOVE/DRAW." The operator then draws the contour which automatically closes itself when the cursor is close enough to the initial starting point. It is possible to draw more than one contour at each probability level. To do this, move the cursor, press "MOVE/DRAW" and draw the contour. Once the contour(s) has been drawn, the button labeled "ENTER" accepts the contours drawn. "ERASE CONTOUR" erases the last contour drawn and "ERASE ALTITUDE" erases all the contours at this probability level.

6. If M = N, go to Step 7, otherwise go to Step 3 with M replaced by M + 1.

Steps 7, 8, 9 similar to Steps 2, 3, 4, 5, 6 except the computer displays "INPUT FINAL CONTOURS" and the final distribution is specified.

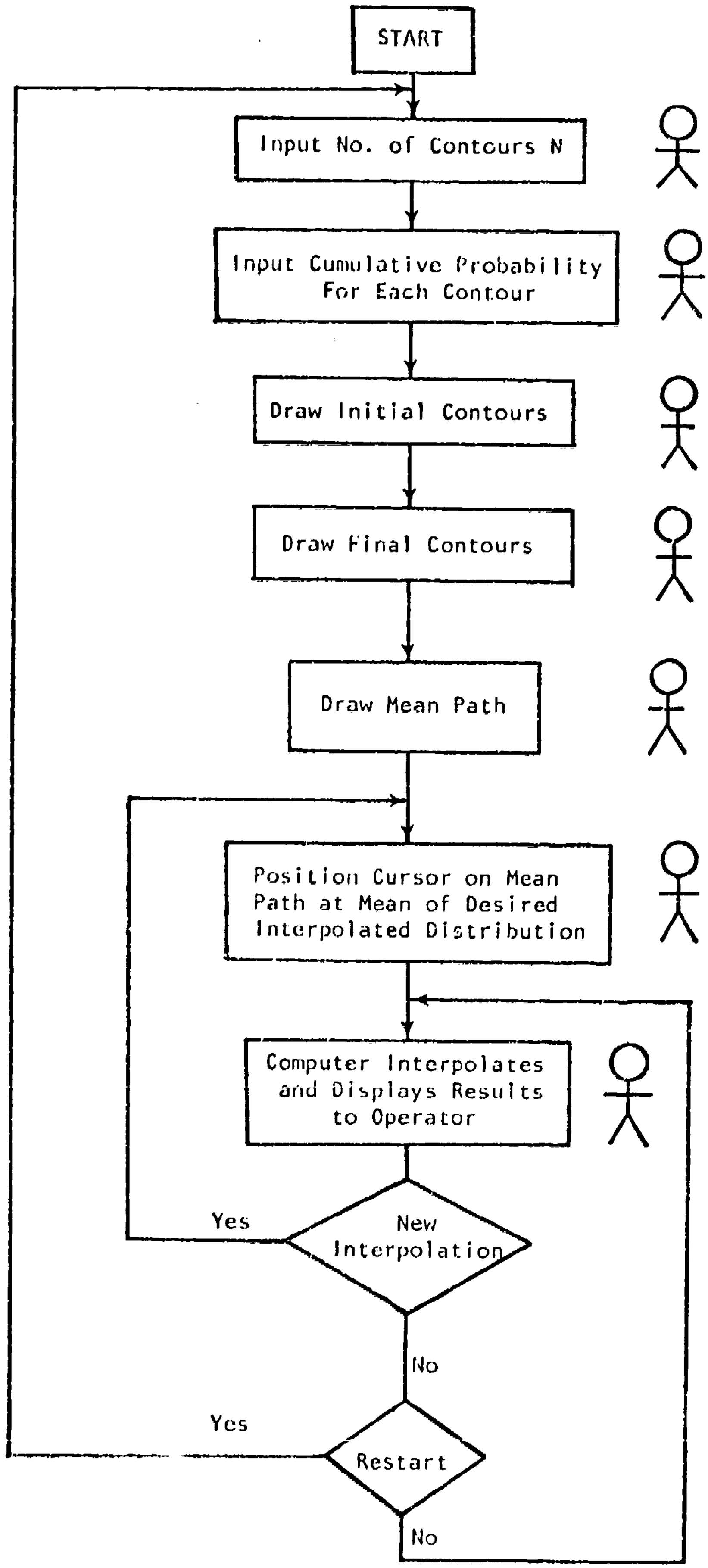


Figure 18. Operator Flowchart for Sampling Program

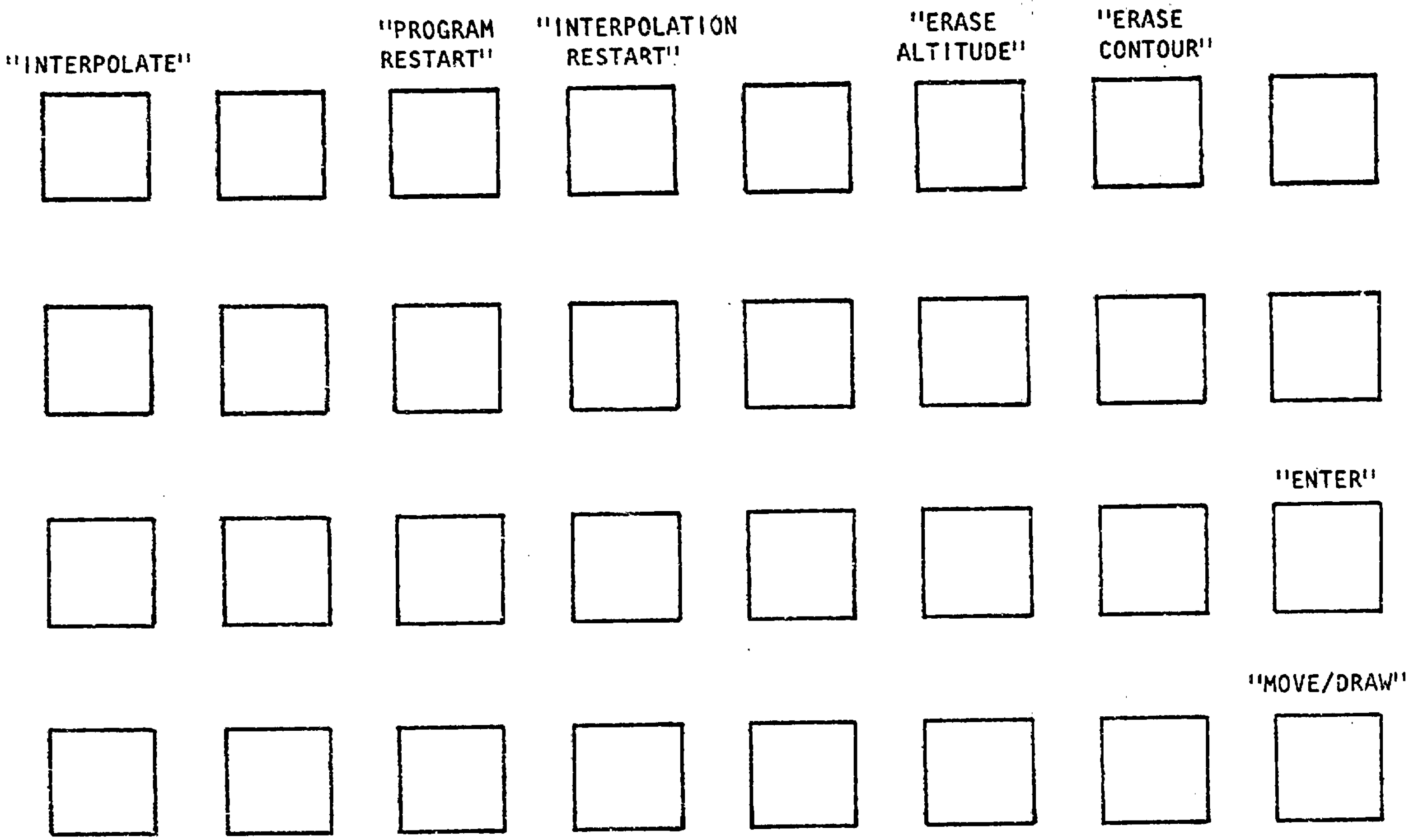


Figure 19. Function Keyboard Layout for the Sampling Program

B. Specify the Mean Path

1. The computer displays "INPUT MEAN PATH" and "MOVE CURSOR."

2. Move the cursor to the mean of the initial distribution.

3. Push "MOVE/DRAW" and draw the mean path.

4. When the mean of the final distribution is reached again push "MOVE/DRAW" then "ENTER."

At this time the display goes blank while the computer samples the initial and final distributions. When this process is completed the display returns with a yellow dot displayed for each sample in the initial and final distribution.

C. Specifying Interpolation Point on Mean Path

1. Press "MOVE/DRAW" and a cursor appears at the start of the mean path.

2. Pressing any button except "INTERPOLATE" moves this cursor along the mean path one segment at a time.

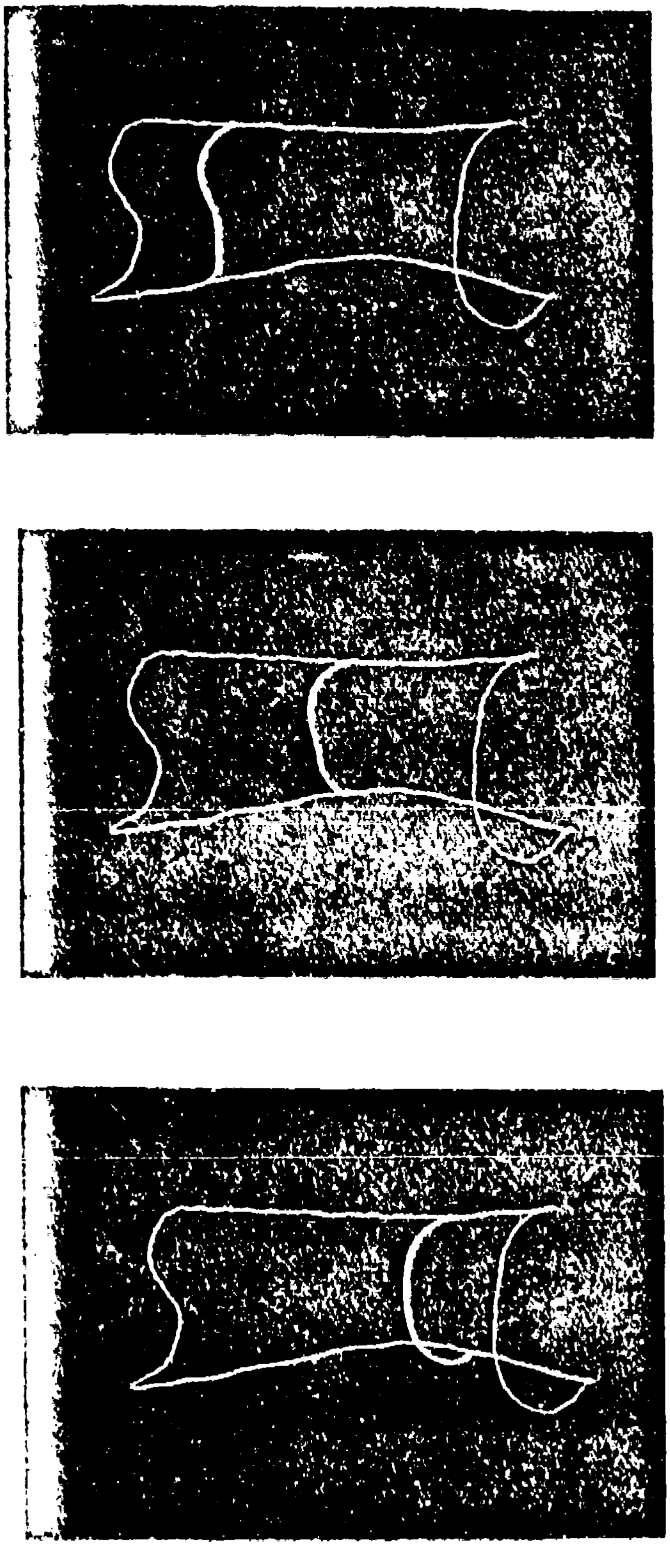
3. When the cursor is at the desired position, press "INTERPOLATE" and an interpolated sample is created. The interpolated sample is again displayed as a group of dots appropriately located between the initial and final distributions.

4. Pushing "INTERPOLATION RESTART" restarts at Step C2, and button "PROGRAM RESTART" restarts the entire program.

5.0 RESULTS

The contour interpolation algorithm and the sampling algorithm described in Sections 3 and 4 were programmed and several aspects of their behavior studied. Figures 20 through 22 contain xerox copies of 35mm color photographs taken of the IDI Display. In Figure 20 we see the contour interpolation algorithm in operation. This figure is divided into three parts showing the initial (S-shaped) contour on the left with the terminal (C-shaped) contour on the right. The two approximately horizontal lines are the designated transition boundaries. Part A of the figure shows an intermediate contour which has evolved to a point approximately 1/4 of the distance between the initial and terminal contours. Parts B and C of the figure contain photographs of the display as the intermediate contour had evolved approximately 1/2 and 3/4 of the way between the initial and terminal contours. It is seen that a more complex figure could be constructed and evolved from segments such as shown in Figure 20. Also it is important to note that the contour interpolation algorithm has the capability of evolving contours which exceed or "lap over" the transition boundaries as demonstrated by the terminal contour. The importance of this feature is discussed below.

In Figure 21 we see the interpolation algorithm being used to uniformly evolve one arbitrary "closed contour" into another arbitrarily shaped closed contour. The interpolation algorithm allows this to be done when the transition boundaries are defined close together and parallel to one another. In this case the separation of the transition boundaries has been exaggerated to show that it is still the same algorithm in operation. The figure shows an intermediate contour at approximatley the half way point. It is the algorithm in this form which would be used for evolving the individual probability contours of NPPDF's between terminating contours which were either operator-generated or obtained through a statistical process such as provided by the sampling algorithm.



(A)

(B)

(C)

Figure 20. Display of Contour Interpolation Algorithm - General Case

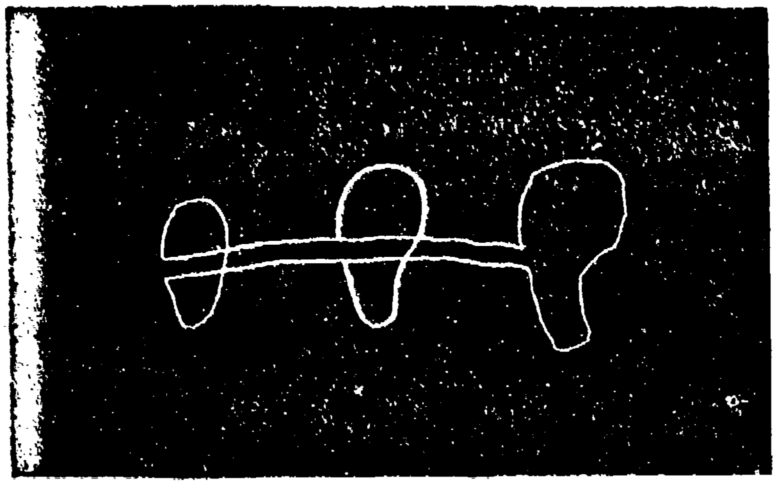


Figure 21. Display of Contour Interpolation Algorithm - Example of "Closed" Contour

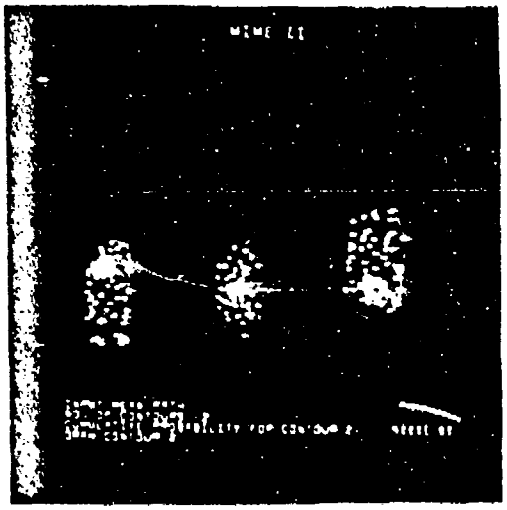


Figure 22. Display of Sampling Algorithm

In Figure 22 we see the operation of the sampling algorithm. The left hand and right hand distributions are denoted by the solid contours. In this case the PDF is represented by two levels of contours--an external contour and a single internal contour. The internal contour is difficult to see in the photograph. However, its location is identifiable from the higher density cluster of sample dots. The line joining the left and right hand contours describes the desired locus of the PDF's central tendency. The center group of dots represent a statistically derived sample of the intermediate NPPDF at approximatley the half way point. We note that the peak of this distribution has also moved approximatley half way between its "high location" in the initial contour to its "low location" in the terminal contour.

The intermediate NPPDF's defined by the sample in Figure 22 would next be represented by a two level closed contour which is of the same topological form as the initial and terminal contours. Thus it is seen that sampling algorithms can be used to generate intermediate NPPDF's on a statistical basis. At this point the contour interpolation algorithm would be used to produce the continuous or closely spaced NPPDF's as the distribution evolves from right to left. We anticipate that using the interpolation algorithm together with the sampling algorithm would provide a computationally efficient method of obtaining a continuous history of the dynamically evolving NPPDF.

6.0 CONCLUSIONS AND RECOMMENDATIONS

This study has produced two new algorithms for developing and processing non-parametric probability distribution functions (NPPDF's) using interactive computer graphics. These methods appear to have potential use in future systems for defense against swimmer attack and other military command and control system applications.

Specifically, two algorithms were developed. The first is a fast, non-statistical method of interpolating generalized contours which is based upon a "minimum order polynomial" method of point-by-point interpolation. It was found that this algorithm is usable in real-time applications with interactive graphics. This algorithm would find use in generating the intermediate contours describing a continuous evolution of NPPDF's between predesignated initial and terminal NPPDF's.

In order to permit the interpolation of NPPDF's along more complex paths, a second algorithm has been developed and demonstrated which will generate a sample of points to define the intermediate NPPDF. From the sample, various algorithms can derive sufficient data to regenerate a contour-like description of the intermediate NPPDF. The resulting NPPDF would then have a measure of statistical validity and could be used as an anchor point for generating the continuously evolving distributions using the interpolation algorithm described above.

It was also found that current generalized techniques for estimating PDF's from a sample are not sufficiently fast for many of the real-time command and control applications foreseen for this technology. Two primary ways could be used to overcome this speed deficiency. The first is to develop new and more rapid PDF estimation techniques that produce usable results in the required time frame; and second, a special PDF estimation processor appears feasible which could use existing algorithms and be instrumented so as to perform parallel computations. Initial estimates show that such a special-purpose hardware using microprocessors could cut down the PDF estimation time by two orders of magnitude over those achieved in computers performing sequential processing.

In order to continue the development of these new techniques, we recommend that the following tasks be performed. These tasks would be divided into two phases.

The first phase should (1) develop more rapid means of generating statistically-based, intermediate PDF's, and (2) evaluate the continuous evolution of PDF's by comparison with the evolution of parametric PDF's of known dynamic properties (e.g., the covariance prediction equation of the Kalman filter).

The next phase should include a human factors experiment performed on an interactive graphics system using tactically significant scenarios. These experiments would be designed and executed for the purpose of testing the ability of human operators to (1) estimate NPPDF's from the types of data available in realistic tactical command and control systems, and (2) predict the predict the evolution of the NPPDF's in a machine-usable form.

APPENDIX A. CONTOUR INTERPOLATION PROGRAM


```
C     MIME1B -- M. SCHECHTERMAN  AUG. 1977
        COMMON/A/IDPL(3500),IERR
        COMMON/ATT/IATT(12)
        COMMON/CONT/IXY(800),NXY0,ICURSR,IMSG,ICOLOR(2),LST,IALT,ILAMP(4),
       *IDMIN,ISTART,IX0,IY0,IEND,IXEND,IYEND,IOFSCR
        COMMON/GRID/AL(2,129),AR(2,129),B(2,129),AU(2,129),AD(2,129),DELP
        COMMON/RECT/DP0(129),SINTH(129),COSTH(129)
        ICURSR=1
        IMSG=2
        IDMIN=20
        IOFSCR=1
        ILAMP(1)=0
        ILAMP(2)=2
        ILAMP(3)=4
        ILAMP(4)=8
 5      CALL GINI(IDPL,3500,IATT,IERR)
        CALL GBEG(ICURSR,0,0)
        CALL GBEG(IMSG,0,-50)
C     DEFINE INITIAL AND FINAL CONTOURS
        IALT=3
        ICOLOR(1)=0
        ICOLOR(2)=0
        LST=0
        ISTART=0
        IEND=0
 10     NXY0=1
        CALL CONTUR
        CALL AINIT
        IALT=IALT+1
        IF (IALT.EQ.4) GO TO 10
C     DEFINE TRANSITION BOUNDARIES
        IX0=AL(1,1)
        IY0=AL(2,1)
        IXEND=AR(1,1)
        IYEND=AR(2,1)
        ICOLOR(1)=1
        ICOLOR(2)=1
        LST=1
        ISTART=1
        IEND=1
 20     NXY0=1
        CALL CONTUR
        CALL AINIT
        IX0=AL(1,129)
        IY0=AL(2,129)
        IXEND=AR(1,129)
        IYEND=AR(2,129)
        IALT=IALT+1
        IF (IALT.EQ.6) GO TO 20
        CALL RECTIL
        CALL MIMOUT
        CALL GLMP(1,-1,0)
        GO TO 5
        END
0 ERRORS COMPILATION COMPLETE
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
      SUBROUTINE AINIT
      COMMON/CONT/INY(200),NNXO,ICURSR,INCG,ICOLOR(2),LST,IALT,ILAMP(4),
     *IDMIN,ISTART,IY0,IX0,IEND,IXENT,IYEND,ICURCK
      COMMON/GRID/AL(2,129),AR(2,129),B(2,129),AU(2,129),AD(2,129),DELP
      DIMENSION X(129),Y(129)
      DIMENSION JNY(200)
C     FIRST ORDER SMOOTHING
      IMAX=NNXO-5
      DO 5 I=1,IMAX
      JNY(I)=INY(I)
    5 CONTINUE
      INY(1)=(5*JNY(1)+2*JNY(3)-JNY(5))/6
      INY(2)=(5*JNY(2)+2*JNY(4)-JNY(6))/6
      IMAX=IMAX-2
      DO 10 I=3,IMAX
      INY(I)=(JNY(I-2)+JNY(I)+JNY(I+2))/3
   10 CONTINUE
      INY(IMAX+1)=(-JNY(IMAX-3)+2*JNY(IMAX-1)+5*JNY(IMAX+1))/6
      INY(IMAX+2)=(-JNY(IMAX-2)+2*JNY(IMAX)+5*JNY(IMAX+2))/6
C     FIND LENGTH OF CONTOUR
      IMAX=IMAX+1
      DALT=0.0
      DO 15 I=3,IMAX,2
      DX=INY(I)-INY(I-2)
      DY=INY(I+1)-INY(I-1)
      DALT=DALT+SQRT(DX*DX+DY*DY)
   15 CONTINUE
C     DIVIDE INTO EQUAL-LENGTH SEGMENTS
      DEL=DALT/128.
      X(1)=INY(1)
      Y(1)=INY(2)
      D=0.0
      J=1
      DO 50 I=2,128
      XIM1=I-1
      DMAX=XIM1*DEL
      IF (D.GE.DMAX) GO TO 35
   20 J=J+2
      DSAVE=D
      DX=INY(J)-INY(J-2)
      DY=INY(J+1)-INY(J-1)
      D=D+SQRT(DX*DX+DY*DY)
      IF (D.LT.DMAX) GO TO 20
   35 TEMP=(DMAX-DSAVE)/(D-DSAVE)
      XSAVE=INY(J-2)
      YSAVE=INY(J-1)
      X(I)=XSAVE+TEMP*DX
      Y(I)=YSAVE+TEMP*DY
   50 CONTINUE
      X(129)=INY(NNXO-2)
      Y(129)=INY(NNXO-1)
      IF (IALT.EQ.1) GO TO 150
      IF (IALT.EQ.2) GO TO 250
      IF (IALT.[illegible]) GO TO 250
      DO 100 I=1,129
      AL(1,I)=X(I)
      AL(2,I)=Y(I)
  100 CONTINUE
      RETURN
  150 DO 200 I=1,129
      AR(1,I)=X(I)
      AR(2,I)=Y(I)
  200 CONTINUE
      RETURN
  250 DO 300 I=1,129
      B(1,I)=X(I)
      B(2,I)=Y(I)
  300 CONTINUE
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
[illegible]                                                    [illegible]                    [illegible]
      [illegible]
      [illegible]    [illegible]
      [illegible]
      [illegible]
      [illegible]  400  CONTINUE
      [illegible]
      [illegible]       RETURN
      [illegible]
0    [illegible] COMPILATION COMPLETE
```

```
      SUBROUTINE RECTIL
      COMMON/GRID/AL(2,129),AR(2,129),B(2,129),AU(2,129),AD(2,129),DELP
      COMMON/RECT/DPO(129),SINTH(129),COSTH(129)
C     COMPUTE TRANSITION LENGTHS, SINES, AND COSINES
C     I.E., L-SUB-N, SINE AND COSINE OF THETA SUB-N
      DO 50 I=1,129
      DX=AU(1,I)-AD(1,I)
      DY=AU(2,I)-AD(2,I)
      DPO(I)=SQRT(DX*DX+DY*DY)
      SINTH(I)=DX/DPO(I)
      COSTH(I)=DY/DPO(I)
   50 CONTINUE
      XPO=AD(1,1)
      YPO=AD(2,1)
      XPN=AR(1,129)
      YPN=AR(2,129)
      DO 100 I=1,129
C     SHIFT AL AND AR TO THE ORIGIN
      X1=AL(1,I)-XPO
      Y1=AL(2,I)-YPO
      X2=AR(1,I)-XPN
      Y2=AR(2,I)-YPN
C     ROTATE AL AND AR, STRETCH AR, MOVE AR TO STRAIGHTEN OUT AP
      AL(1,I)=X1*COSTH(1)-Y1*SINTH(1)
      AL(2,I)=X1*SINTH(1)+Y1*COSTH(1)
      AR(1,I)=X2*COSTH(129)-Y2*SINTH(129)+128.*DELP
      AR(2,I)=(X2*SINTH(129)+Y2*COSTH(129))*DPO(1)/DPO(129)
  100 CONTINUE
C     NEW RIGHT HAND SIDE MUST BE COMPUTED WITH EQUAL LENGTH SEGMENTS
      D=0.0
      DO 150 I=1,128
      DX=AR(1,I+1)-AR(1,I)
      DY=AR(2,I+1)-AR(2,I)
      D=D+SQRT(DX*DX+DY*DY)
  150 CONTINUE
      DEL=D/128.
      D=0.0
      J=0
      DO 300 I=2,128
      XNNI=I-1
      DMAX=XNNI*DEL
      IF (D.GE.DMAX) GO TO 250
  200 J=J+1
      DOLD=D
      DX=AR(1,J+1)-AR(1,J)
      DY=AR(2,J+1)-AR(2,J)
      D=D+SQRT(DX*DX+DY*DY)
      IF (D.LT.DMAX) GO TO 200
  250 TEMP=(DMAX-DOLD)/(D-DOLD)
      XSAVE=AR(1,J)
      YSAVE=AR(2,J)
      B(1,I)=XSAVE+TEMP*DX
      B(2,I)=YSAVE+TEMP*DY
  300 CONTINUE
      DO 350 I=2,128
      AR(1,I)=B(1,I)
      AR(2,I)=B(2,I)
  350 CONTINUE
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE MIMOUT
      COMMON/ATT/IATT(12)
      COMMON/GRID/AL(2,129),AR(2,129),B(2,129),AU(2,129),AD(2,129),DELP
      COMMON/VECT/[illegible],SINTH(129),COSTH(129)
      IBOUND=1
      JBOUND=1
      CALL GLMP(1,5,1)
      CALL GLMP(1,10,1)
      CALL GLMP(1,10,1)
      CALL GPEN(7,0,0)
      CALL GPUT(6,1760,0,1)
 150  DO 200 J=1,129
      CALL GCYT(0)
      IATT(1)=0
      CALL GSTT(0,0)
      CALL UPDATE(J)
      IEL=7
      IX=B(1,1)
      IY=B(2,1)
      CALL GPUT(IEL,73,IX,IY)
      IEL=IEL+1
      DO 200 I=2,129
      IX=B(1,I)
      IY=B(2,I)
      CALL GPUT(IEL,53,IX,IY)
      IEL=IEL+1
 200  CONTINUE
      CALL GHLT
      CALL GCYT(1)
      DO 275 NCYC=1,5
      CALL INTRPT
      IF (IATT(1).NE.36 .OR. IATT(5).NE.-1) GO TO 275
      IF (IATT(3).EQ.8) GO TO 250
      IF (IATT(3).EQ.12) GO TO 265
      IF (IATT(3).NE.13) RETURN
      IF (IBOUND.EQ.0) GO TO 240
      IBOUND=0
      CALL GEOF(5)
      CALL GEOF(6)
      GO TO 275
 240  IBOUND=1
      CALL GEON(5)
      CALL GEON(6)
      [illegible]
 250  IF (JBOUND.EQ.0) GO TO 260
      JBOUND=0
      CALL GEOF(3)
      CALL GEOF(4)
      GO TO 275
 260  JBOUND=1
      CALL GEON(3)
      CALL GEON(4)
      GO TO 275
 265  CALL INTRPT
      IF (IATT(1).NE.36 .OR. IATT(3).NE.12) GO TO 265
      IF (IATT(5).EQ.0) GO TO 265
 275  CONTINUE
 290  CONTINUE
      GO TO 150
      END
0 ERRORS COMPILATION COMPLETE
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
          SUBROUTINE ENTRPT
          COMMON/[illegible]/[illegible](12)
          [illegible](1)=2
          CALL GETT(0,0)
    1     CONTINUE
          IF ([illegible](1).EQ.0) GO TO 1
          RETURN
          END
  0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE UPDATE(J)
      COMMON/GRID/[illegible]
      [illegible]
      DO 100 I=1,[illegible]
      [illegible]
C     INTERPOLATE, MOVE TO ORIGIN, UNROTATE [illegible]
      [illegible]
      [illegible]
C     UNROTATE AND MOVE TO [illegible] POSITION
      [illegible]
      [illegible]
  100 CONTINUE
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
/REW,14
/ASSIGN,PT,14
.NEM,12
/PORT,M,N,F,H
```

```
      SUBROUTINE CONTUR
      COMMON/ATT/IATT(12)
      COMMON/CONT/INXY(500),NXY0,ICURSR,IMSG,ICOLOR(2),LST,IALT,ILAMP(4),
     XIDMIN,ISTART,IX0,IY0,IEND,IXEND,IYEND,IOFSOR
      DMIN2=IDMIN*IDMIN
C  DEFINE CURSOR
      CALL GEON(ICURSR)
      CALL GENT(ICURSR)
      IF (ISTART.NE.0) GO TO 20
      IX0=0
      IY0=0
 20   CALL GPUT(6,72,0,0)
      CALL GPUT(7,1000,8,0)
      CALL GPUT(8,1000,5,0)
C  DEFINE SUBJECT CONTOURS
      CALL GBEG(IALT,IX0,IY0)
      CALL GBEG(3,120,1,LST)
      CALL GPUT(4,140,5,ICOLOR(1))
      CALL GPUT(4,140,0,ICOLOR(2))
      IF (IOFSOR.NE.0) CALL GPUT(5,1740,0,1)
C  DEFINE MESSAGE AREA
      CALL GCHA(IMSG,6,0,0,12)
C  ALLOW SUBJECT TO DRAW IN CONTOURS
 290  CONTINUE
C  NXY -- SUBSCRIPT FOR INXY
      NXY=NXY0
C  IPTR1 -- PTR TO STARTING ELEMENT FOR CURRENT CONTOUR
      IPTR1=0
      IXP=IX0
      IYP=IY0
      IEX=0
      IEY=0
      IEL=0
      CALL GCYT(1)
      IF (ISTART.NE.0) GO TO 400
      IXT=0
      IYT=0
C  MOVE CURSOR TO START POSITION
 300  CALL GEON(IMSG)
      CALL GULT
      CALL [illegible](IMSG,0)
      WRITE(11,1001)
 1001 FORMAT('MOVE CURSOR [illegible]')
      CALL GLMP(1,ILAMP(1),1)
      [illegible]
      CALL [illegible]
      [illegible] GLMP(1,ILAMP(3),1)
      CALL GLMP(1,ILAMP(4),1)
 200  CALL [illegible]
      [illegible] 220
      [illegible]
      [illegible] 200
      [illegible] 300
      [illegible]
      [illegible] 300
      [illegible]
C  [illegible]
 220  CALL [illegible]
      CALL [illegible],IX,IY)
      GO TO 200
 240  [illegible] 200
C  [illegible]
      [illegible]
C  [illegible] LESS THAN IDMIN RASTER UNITS LONG
      [illegible]
 24?  CALL [illegible]
      [illegible]
      [illegible]
```

```
      NXN=IPTR2
      CALL GLMP(1,ILAMP(3),0)
      [illegible]
      IAY=LOY
      [illegible]
      [illegible]
      IF(IEL.NE.3)GOTO 308
      CALL GLMP(1,ILAMP(2),0)
      CALL GLMP(1,ILAMP(4),0)
      IF (ISTART.NE.0) GO TO 400
      GOTO 208
C  ERASE ENTIRE ALTITUDE
 350  IF(IEL.[illegible].3)GOTO 200
      CALL GLMP(1,ILAMP(2),0)
      CALL GLMP(1,ILAMP(3),0)
      CALL GLMP(1,ILAMP(4),0)
      CALL SENT(IALT)
C  IEL MAY BE A VECTOR LESS THAN IDMIN RASTER UNITS LONG
      [illegible]
 302  CALL SPOT(IX,IY,0,0)
      GOTO 200
C  AUTOMATIC STARTING POINT
 400  IX=IXO
      IY=IYO
      CALL GLMP(1,ILAMP(1),1)
      CALL TRACK(IXT,IYT)
C  IXP,IYP -- LAST POINT ON PREVIOUS CONTOUR
C  IXS,IYS -- START POSITION OF CURRENT CONTOUR
C  IXL,IYL -- LAST POSITION OF CURRENT CONTOUR
C
C  START CONTOUR
 500  CALL CHCT
      CALL GLCH(IPEO,0)
      WRITE(15,1002)
 1002 FORMAT('DRAW CONTOUR')
      CALL GLMP(1,ILAMP(1),0)
      CALL GLMP(1,ILAMP(3),0)
      CALL GLMP(1,ILAMP(4),0)
      CALL SENT(IALT)
      IXL=IX
      IYL=IY
C  IDX,IDY -- USED TO REPOSITION PEN TO WHERE CURSOR IS,  OTHERWISE
C             SUBJECT CONTOUR AND CURSOR POSITION WON'T MATCH.
      IDX=IXL-[illegible]
      IDY=IYL-[illegible]
      CALL [illegible](IX,IY,IDX,IDY)
      IXP=IX
      IYP=IY
      [illegible]
C  IPTR2 -- PTR TO START OF CURRENT CONTOUR IN INV
      [illegible]
      [illegible]
      IF(NXN.GT.[illegible])GOTO 9[illegible]
      [illegible]=IXL
      [illegible]=IYL
      NXN=NXN+2
 510  CALL TRACK(IX,IY)
      IX=[illegible]
      IY=IY-[illegible]
      IF(IPTR(1).NE.0)GOTO 520
      IF(IPTR(5).NE.0)GOTO 510
      IF(IPTR(2).NE.ILAMP(1))GOTO 510
C
C  END CONTOUR
 550  CONTINUE
      IF([illegible].NE.1)GOTO 551
C  [illegible] TO MOVE CURSOR
      CALL SENT(IALT)
```

```
      CALL CPUT(IPTRI,90,0,0)
C     IEL MAY BE A VECTOR LESS THAN 10MIN RASTER UNITS LONG
      CALL GPUT(IEL,73,0,0)
      [illegible]
      [illegible]
      [illegible]
      IF (IEL.NE.0 .AND. ISTART.NE.0) GO TO 400
      GOTO [illegible]
  551 CONTINUE
C     DRAW VECTOR
      IXL=IX
      IYL=IY
      IEL=[illegible]
      IF (NXY.GT.500) GO TO 9000
      IXY(NXY)=IX
      IXY(NXY+1)=IY
      NXY=NXY+2
C     CHECK FOR AUTOMATIC CLOSING
      IF (ISNB.EQ.0 .OR. IPTRI.NE.0) GO TO 552
      IDX=IXEND-IXL
      IDY=IYEND-IYL
      GO TO 553
  552 IDX=IXT-IXL
      IDY=IYT-IYL
  553 DX=IDX
      DY=IDY
      DL=SQRT(DX*DX+DY*DY)
      IF(DL.LT.DMIN2)GOTO 554
      IF(NXY.GT.500)GOTO 9000
C     WRITE END OF CONTOUR MARKER INTO IXY ARRAY
      IXY(NXY)=2000
      IXY(NXY+1)=2000
      NXY=NXY+2
      IXN=0
      IYN=0
      GO TO 557
C
C     AUTOMATIC CLOSURE
  554 CONTINUE
      CALL [illegible]
      CALL GPUT(IEL,52,IDX,IDY)
      [illegible]
      [illegible]
      [illegible]
      IF(NXY-2.GT.500)GOTO 9000
      IF (ISNB.EQ.0 .OR. IPTRI.NE.0) GO TO 555
      [illegible]
      [illegible]
      [illegible]
  [illegible] [illegible]
      [illegible]
  [illegible] [illegible]
      [illegible]
      [illegible]
  557 [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      GOTO [illegible]
C
C     [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
```

THIS PAGE IS BEST QUALITY PRACTICABLE FROM COPY FURNISHED TO DDC

```
      DY=IDY
      DS=DX*DX+DY*DY
      CALL CENT(IELT)
      CALL UPUT(IEL,SC,IDX,IDY)
      IF(DS.LT.DHIH2)GOTO 510
C   DRAW VECTOR
      IXL=IX
      IYL=IY
      ICL=ICL+1
      IF(NXY.GT.800)GOTO 2000
      IXY(NXY)=IX
      IXY(NXY+1)=IY
      NXY=NXY+2
C   CHECK FOR AUTOMATIC CLOSING
      IF (IEND.EQ.0 .OR. IPTRI.NE.0) GO TO 540
      IDX=IXEND-IXL
      IDY=IYEND-IYL
      GO TO 545
 540  IDX=IXF-IXL
      IDY=IYF-IYL
 545  DX=IDX
      DY=IDY
      DS=DX*DX+DY*DY
      IF(DS.GT.DHIH2)GOTO 510
      GOTO 5[illegible]4
C
C   EXIT ROUTINE
 730  CONTINUE
      IF(NXY.GT.800)GOTO 2000
      [illegible]
      CALL CLMP([illegible])
      CALL CLMP([illegible])
C   [illegible] INTO IXY ARRAY
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
      [illegible]
 2000 [illegible]
 1002 [illegible] NO MORE ROOM IN STORAGE VECTOR IXY')
      [illegible]
      [illegible]
0 ERRORS, COMPILATION COMPLETE
```

```
PAGE    1                       VORTXII    FTN IV(G)              0003 HOURS
          SUBROUTINE TRACKB(IX,IY)
          COMMON/ATT/IATT(10)
C  FIND TRACKBALL POSITION
C  RESTRICTIONS:
C     CYCLE TIMER ALREADY TURNED ON
C  CAUTION:
C     ALL OTHER INTERRUPTS ARE IGNORED EXCEPT THOSE FROM KEYBOARD,
C     WHICH CAUSE IMMEDIATE RETURN
C     IF KEYBOARD INTERRUPT OCCURRED, IX AND IY ARE UNCHANGED
C
  300     IATT(1)=0
          CALL SSTT(0,0)
  310     CONTINUE
          IF (IATT(1).EQ.0) GO TO 310
          IF (IATT(1).EQ.30) RETURN
          IF (IATT(1).NE.20) GO TO 300
          CALL CNLV(1,IX,IY)
          IX=IX/5+500
          IY=IY/5+500
          RETURN
          END
 0 ERRORS COMPILATION COMPLETE
```

```
TIDB,MIME1B,2,0
[illegible]
[illegible]
[illegible]   A 77150   V$JPPF  A 77107   V$[illegible] A 77035   V$[illegible] A 77[illegible]
[illegible]   A 77003   HSCOMB  A 75[illegible]  [illegible] A [illegible]   V$[illegible] A [illegible]
[illegible]   A [illegible]   [illegible] A [illegible]   [illegible] A [illegible]   V$[illegible] A [illegible]
V$[illegible] A 70641   V$ALTB  A [illegible]   [illegible] A 54160   [illegible] A [illegible]
[illegible]   A 54150   [illegible] A [illegible]   [illegible] A [illegible]   [illegible] A [illegible]
[illegible]   A [illegible]   [illegible] A [illegible]   V$[illegible] A [illegible]   V$[illegible] A [illegible]
V$[illegible] A 54073   [illegible] A [illegible]   [illegible]   [illegible]   [illegible]
V$DISP  A 65053   DISP1   A 65050   V$SCPT  A 65034   V$STUD  A 55005
V$TYH   A 63171   IDIION  A 63152   I2FGOA  A [illegible]   V$MTH   A 61746
V2DB    A 57776   TIDBOC  A 76261   TIPBJP  A 76227   CTTYOA  A 76201
CTOROH  A 76014   CTPTOH  A 75757   CTLPOE  A 75557   CTMTOH  A 75534
CTDOD   A 75272   CTWCS   A 75265   V$WCS   A 75252   V$WCSE  A 75242
S$FCB   A 77242   SIFCB   A 77160   PIFCB   A 77172   LOFCB   A 77204
GOFCB   A 77254   BOFCB   A 77230   BIFCB   A 77216   V$REPF  R 35450
V$IOST  A 74733   V$IOC   A 73041   V$EXEC  A 71424   V$DSTB  A 54133
POFCB   A 77266   $EE     R 43665   W$X     R 35043   V$OPNB  R 31630
V$OPEN  R 34637   V$CLSB  R 344[illegible]   V$[illegible] R [illegible]   P$X     R 35034
[illegible]
```

APPENDIX B. SAMPLING PROGRAM

```
PAGE    1              MIMEII     VORTXII    FTN IV(G)            0009 HOURS
C**************************************************************************************
C
C        MIME II-18 JULY 1977
C
C**************************************************************************************
         COMMON /D/IDPL(8000),IERR
         COMMON /ATT/IATT(12)
         COMMON /MIME2/NCNTUR,PROBC(5),NRIPC(5,3),NIXY(5,3),NRPATH,NIPATH,
        1              IXMIN(2),IXMAX(2),IYMIN(2),IYMAX(2)
         COMMON /COMUN/NOVPTR,NMOVLY(20),IFLAG(20)
         COMMON /CONTUR/NXY,IXY(800),ICOLOR(2),LST,ICURSR,IALT,INSG
         COMMON /SAM/IS(2,100,3),NI(5,3)
         COMMON /DISK/IFCB(13)
         COMMON /MISS/NMS
         INTEGER OVLYNM(3,4)
         DATA OVLYNM(1,1),OVLYNM(2,1),OVLYNM(3,1)/2HIN,2HPU,2HT /
         DATA OVLYNM(1,2),OVLYNM(2,2),OVLYNM(3,2)/2HCO,2HNT,2H2 /
         DATA OVLYNM(1,3),OVLYNM(2,3),OVLYNM(3,3)/2HSA,2HMP,2HLE/
         DATA OVLYNM(1,4),OVLYNM(2,4),OVLYNM(3,4)/2HMI,2HNE,2H00/
         DATA KEYMAX/4/
         CALL V$OPEN(18,15,IFCB,0)
      10 CALL GINI(IDPL,8000,IATT,IERR)
         CALL GBEG(1,400,1023)
         CALL GCHA(1,6,0,3,7)
         CALL GPUT(6,1750,3,1)
         CALL IZAP(IFLAG,20,0)
         WRITE(15,1000)
         CALL GSTT(0,0)
         KEY=1
      30 IF(KEY.GT.KEYMAX) GO TO 999
         CALL OVLAY(0,0,OVLYNM(1,KEY))
      40 NOVPTR=NOVPTR+1
      50 KEY=NMOVLY(NOVPTR)
         IF(KEY.GT.0) GO TO 30
         IF(KEY.EQ.0) GO TO 10
   C KEY IS NEGATIVE FOR LOOPS
   C I.E. NMOVLY=4,5,6,-2 WORKS LIKE NMOVLY=4,5,6,5,6,5,6,5,......
         NOVPTR=-KEY
         GO TO 50
     999 WRITE(5,1001)
         GO TO 10
    1000 FORMAT(7HMIME II)
    1001 FORMAT(' ERROR-OVERLAY NUMBER TOO LARGE')
         END
 0 ERRORS COMPILATION COMPLETE
```

```
PAGE    1              MINEII    VORTXII    FTN IV(G)              0009 HOURS
      C
      C
            BLOCK DATA
            COMMON /DISK/IFCB(13)
            DATA IFCB(3),IFCB(8),IFCB(9),IFCB(10)/0,2HT1,2H  ,2H  /
            END
  0 ERRORS COMPILATION COMPLETE
 /WEOF,14
 /MEM,6
 /FORT,M,H,F
```

```
      SUBROUTINE INPUT
      COMMON /D/IDPL(8000),IERR
      COMMON /MINE2/NCNTUR,PROBC(5),NRIPC(5,3),NIXY(5,3),NRPATH,NIPATH,
     1              IXMIN(2),IXMAX(2),IYMIN(2),IYMAX(2)
      COMMON /COMUN/NOVPTR,NMOVLY(20),IFLAG(20)
      COMMON /CONTUR/NXY,IXY(800),ICOLOR(2),LST,ICURSR,IALT,IMSG
      COMMON /DISK/IFCB(13)
      COMMON /MISS/NRS
      EQUIVALENCE (ICNTUR,IFLAG(2)),(IELMA,IFLAG(3)),(IELMB,IFLAG(4)),
     1           (IELMC,IFLAG(5)),(IELMD,IFLAG(6)),(IELME,IFLAG(7))
      DATA MAXTUR/5/
      CALL GHLT
      WRITE(5,4040)
 4040 FORMAT(' ENTERED INPUT')
      IF(IFLAG(1).EQ.1) GO TO 150
      IF(IFLAG(1).EQ.-1) GO TO 250
      IF(IFLAG(1).EQ.-2) GO TO 350
      CALL IZAP(IXMIN,2,2000)
      CALL IZAP(IXMAN,2,0)
      CALL IZAP(IYMIN,2,2000)
      CALL IZAP(IYMAX,2,0)
      IFLAG(1)=1
      ICURSR=3
      IMSG=4
      IALT=4
C  DEFINE CURSOR
      CALL GBEG(ICURSR,0,0)
      CALL GPUT(6,73,0,0)
      CALL GPUT(7,1600,8,0)
      CALL GPUT(8,1600,5,0)
      CALL GEOF(ICURSR)
C  DEFINE MESSAGE AREA
      CALL GREG(IMSG,700,0)
      CALL GCHA(IMSG,5,0,2,12)
      CALL GBEG(2,0,0)
      CALL GEOF(IMSG)
      IELM=5
      CALL GHLT
      IELMA=IELM
      CALL SETUPW(2,IELM,0,0,22,2)
      WRITE(15,1000)
      CALL SETUPW(2,IELM,0,-30,17,2)
      WRITE(15,1001)
      IELMB=IELM
   50 IELM=IELMB
      CALL SETUPR(2,IELM,1,2)
      READ(15,2000) NCNTUR
      IF(NCNTUR.GT.MAXTUR) GO TO 50
      NOVPTR=0
      NMOVLY(1)=2
      NMOVLY(2)=1
      NMOVLY(3)=-1
      ICNTUR=0
      IELMC=IELM
  100 IELM=IELMC
      ICNTUR=ICNTUR+1
      IALT=IALT+1
      CALL SETUPW(2,IELM,0,-60,37,2)
      WRITE(15,1002) ICNTUR
      IELMD=IELM
      CALL SETUPR(2,IELM,12,2)
      READ(15,2001) PROBC(ICNTUR)
      IELME=IELM
      CALL SETUPW(2,IELM,0,-90,14,2)
      WRITE(15,1003) ICNTUR
      RETURN
  150 NIXY(ICNTUR,1)=NXY
      NRIPC(ICNTUR,1)=IFCB(4)
```

```
      WRITE(15) (INY(I),I=1,NNY)
      NNY2=NNY/2
      DO 170 I=1,NNY2
      J1=I+I-1
      J2=I+I
      IF(INY(J1).GE.2000) GO TO 170
      IXMIN(1)=MIN0(IXMIN(1),INY(J1))
      IXMAX(1)=MAX0(IXMAX(1),INY(J1))
      IYMIN(1)=MIN0(IYMIN(1),INY(J2))
      IYMAX(1)=MAX0(IYMAX(1),INY(J2))
  170 CONTINUE
      CALL GHLT
      WRITE(5,1099) ICNTUR,NRTPC(ICNTUR,1),IXMIN(1),IXMAX(1),
     1              IYMIN(1),IYMAX(1)
 1099 FORMAT(6I10)
      WRITE(5,1100) (INY(I),I=1,NNY)
 1100 FORMAT(12I10)
      IF(ICNTUR.EQ.NCNTUR) GO TO 199
      GO TO 100
  199 CONTINUE
      CALL GHLT
      ILLAB(1)=-1
      WRITE(5,1230) ICNTUR,NCNTUR,ILLN
 1230 FORMAT(3I6)
      WRITE(5,1230) ICNTUR
      CALL GSCHC(2,ILLMAT(2))
      WRITE(15,1010)
      ICNTUR=0
  200 ICNTUR=ICNTUR+1
      IALT=IALT+1
      CALL GSCHC(2,ILLMC(2))
      WRITE(15,1000) ICNTUR
      WRITE(5,1230) ICNTUR,NCNTUR,ILLN
      CALL GSCHC(2,ILLMP)
      WRITE(15,2001) PRCRC(ICNTUR)
      CALL GSCHC(2,ILLMI(2))
      WRITE(15,1000) ICNTUR
      RETURN
  250 NNY=ICNTUR(2) NNY
      NRTPC(ICNTUR,2)=IIC(B(4))
      WRITE(15) (INY(I),I=1,NNY)
      NNY2=NNY/2
      DO 270 I=1,NNY2
      J1=I+I-1
      J2=I+I
      IF(INY(J1).GE.2000) GO TO 270
      IXMIN(2)=MIN0(IXMIN(2),INY(J1))
      IXMAX(2)=MAX0(IXMAX(2),INY(J1))
      IYMIN(2)=MIN0(IYMIN(2),INY(J2))
      IYMAX(2)=MAX0(IYMAX(2),INY(J2))
  270 CONTINUE
      IF(ICNTUR.EQ.NCNTUR) GO TO 299
      GO TO 200
  299 CONTINUE
C MEAN PATH
      ILLAB(1)=-2
      CALL GSCHC(2,ILLAB(2))
      WRITE(15,1050)
      IALT=IALT+1
      RETURN
  350 NPATH=NNY
      NPATH=IIC(B(4))
      WRITE(15) (INY(I),I=1,NNY)
      NNY=IIC(B(1))
      ILLAB(1)=0
C 5!  GO BACK TO CONTINUE
      WRITE(5,450) NOVPTR,NRNVLY(1)
  450 FORMAT(2I10)
```

```
            MOVPTR=0
            NMOVLY(1)=3
            RETURN
       1000 FORMAT(22HINPUT INITIAL CONTOURS)
       1001 FORMAT(17HNO. OF CONTOURS =)
       1002 FORMAT(35HCUMULATIVE PROBABILITY FOR CONTOUR ,I1,1H=)
       1003 FORMAT(13HDRAW CONTOUR ,I1)
       1010 FORMAT(22HINPUT FINAL CONTOURS   )
       1020 FORMAT(22HINPUT MEAN PATH        )
       2000 FORMAT(I1)
       2001 FORMAT(E12.4)
            END
 0 ERRORS COMPILATION COMPLETE
/WEOF,14
/MEM,6
/FORT,M,H,F
```

```
*********************
*  OVERLAY CONT2
*********************
      SUBROUTINE CONT2
      COMMON /ATT/IATT(12)
      COMMON /CONTUR/NXY,IXY(200),ICOLOR(2),LST,ICURSR,IALT,IMSG
      CALL GHLT
      WRITE(5,4040)
 4040 FORMAT(' ENTERED CONT2')
      CALL GEON(ICURSR)
      CALL GEON(IMSG)
C
C  DEFINE SUBJECT CONTOURS
      CALL GBEG(IALT,0,2)
      CALL GBEG(3,130,1,LST)
      CALL GPUT(4,140,5,ICOLOR(1))
      CALL GPUT(4,140,6,ICOLOR(2))
C  ALLOW SUBJECT TO DRAW IN CONTOURS
  290 CONTINUE
C  NXY -- SUBSCRIPT FOR IXY
      NXY-1
C  IPTR1 -- PTR TO STARTING ELEMENT FOR CURRENT CONTOUR
      IPTR1=0
      IXP=0
      IYP=0
      IAX=0
      IAY=0
      IEL=6
      CALL GCYT(1)
C  MOVE CURSOR TO START POSITION
  305 CALL GEON(IMSG)
      CALL GSCH(IMSG,5)
      WRITE(15,307)
  307 FORMAT('MOVE CURSOR ')
      CALL GLMP(1,0,1)
      IF(IEL.EQ.6)GOTO 308
      CALL GLMP(1,8,1)
      IF(IPTR1.NE.0)CALL GLMP(1,25,1)
      CALL GLMP(1,26,1)
  308 CALL TRACKB(IX,IY)
      IF(IATT(1).NE.26)GOTO 450
      IF(IATT(5).NE.0)GOTO 308
      IF(IATT(3).EQ.0)GOTO 500
      IF(IATT(3).EQ.25)GOTO 346
      IF(IATT(3).EQ.26)GOTO 350
      IF(IATT(3).NE.8)GOTO 308
      IF(IEL-6)308,308,750
  346 IF(IPTR1.EQ.0)GOTO 308
C  ERASE LAST CONTOUR
      CALL GENT(IALT)
C  IEL MAY BE A VECTOR LESS THAN 20 R.U.
      DO 348 IJ=IPTR1,IEL
  348 CALL GPUT(IJ,73,0,0)
      IEL=IPTR1
      IPTR1=0
      NXY=IPTR2
      CALL GLMP(1,25,0)
      IAX=LAX
      IAY=LAY
      IXP=LXP
      IYP=LYP
      IF(IEL.NE.6)GOTO 308
      CALL GLMP(1,8,0)
      CALL GLMP(1,26,0)
      GOTO 308
C  ERASE CURRENT ALTITUDE
  350 IF(IEL.EQ.6)GOTO 308
      CALL GLMP(1,8,0)
```

```
      CALL GLMP(1,25,0)
      CALL GLMP(1,26,0)
      CALL GENT(IALT)
C  IEL MAY BE A VECTOR LESS THAN 20 R.U.
      DO 352 IJ=6,IEL
  352 CALL GPUT(IJ,73,0,0)
      GOTO 290
C  IXP,IYP -- LAST POINT ON PREVIOUS CONTOUR
C  IXF,IYF -- START POSITION OF CURRENT CONTOUR
C  IXL,IYL -- LAST POSITION OF CURRENT CONTOUR
C
C  UPDATE CURSOR POSITION
  450 CALL GENT(ICURSR)
      CALL GPUT(6,73,IX,IY)
      GOTO 308
C  START CONTOUR
  500 CALL GSCH(IMSG,5)
      WRITE(15,502)
  502 FORMAT('DRAW CONTOUR')
      CALL GLMP(1,8,0)
      CALL GLMP(1,25,0)
      CALL GLMP(1,26,0)
      CALL GENT(IALT)
      IXL=IX
      IYL=IY
C  ATTACH START POINT TO VERTICAL EDGES IF CLOSE ENOUGH
      IF(1000-IXL.LE.30)IXL=1000
      IF(IXL.LE.30)IXL=-1
C  IAX,IAY -- USED TO REPOSITION BEAM TO WHERE CURSOR IS;  OTHERWISE
C                SUBJECT CONTOUR AND CURSOR POSITION WON'T MATCH.
      IDX=IXL-IXP+IAX
      IDY=IYL-IYP+IAY
      CALL GPUT(IEL,73,IDX,IDY)
      IXF=IXL
      IYF=IYL
      IPTR1=IEL
C  IPTR2 -- PTR TO START OF CURRENT CONTOUR IN IXY
      IPTR2=NXY
      IEL=IEL+1
      IF(NXY.GT.800)GOTO 9000
      IXY(NXY)=IXL
      IXY(NXY+1)=IYL
      NXY=NXY+2
  510 CALL TRACKB(IX,IY)
      IF(IATT(1).NE.36)GOTO 680
      IF(IATT(5).NE.0)GOTO 510
      IF(IATT(3).NE.0)GOTO 510
C  END CONTOUR
  550 CONTINUE
      IF(IEL-IPTR1.NE.1)GOTO 551
C  IF NO VECTOR DRAWN, GO BACK TO MOVE CURSOR.
      CALL GENT(IALT)
      CALL GPUT(IPTR1,73,0,0)
C  IEL MAY BE A VECTOR LESS THAN 20 R.U.
      CALL GPUT(IEL,73,0,0)
      IEL=IPTR1
      IPTR1=0
      NXY=IPTR2
      GOTO 305
  551 CONTINUE
      JGO=2
      GOTO 680
  549 IDX=IXF-IXL
      IDY=IYF-IYL
      DX=IDX
      DY=IDY
      DS=DX*DX+DY*DY
      IF(DS.LT.900.)GOTO 552
```

```
C  ATTACH OPEN CURVE TO A VERTICAL SCREEN EDGE IF CLOSE ENOUGH
       IF(1000-IXL.GT.30)GOTO 553
       IJ=1000
       GOTO 557
  553  IF(IXL.GT.30)GOTO 554
       IJ=-1
  557  CALL GENT(IALT)
       CALL GPUT(IEL,53,IJ-IXY(NXY-2),0)
       IF(NXY.GT.800 )GOTO 9000
       IXY(NXY)=IJ
       IXY(NXY+1)=IXY(NXY-1)
       NXY=NXY+2
       IXL=IJ
       IEL=IEL+1
  554  IF(NXY.GT.800)GOTO 9000
C  WRITE END OF CONTOUR ONTO OUTPUT FILE
       IXY(NXY)=2000
       IXY(NXY+1)=2000
       NXY=NXY+2
       GOTO 555
  552  CONTINUE
       CALL GENT(IAL )
       CALL GPUT(IEL,53,IDX,IDY)
       IEL=IEL+1
       LAX=IAX
       LAY=IAY
       IAX=-IDX
       IAY=-IDY
       IF(NXY+2.GT.800)GOTO 9000
       IXY(NXY)=IXP
       IXY(NXY+1)=IYP
       IXY(NXY+2)=2000
       IXY(NXY+3)=2000
       NXY=NXY+4
       GOTO 556
  555  LAX=IAX
       LAY=IAY
       IAX=0
       IAY=0
  556  LXP=IXP
       LYP=IYP
       IXP=IXL
       IYP=IYL
       GOTO 305
C  UPDATE CURSOR POSITION
  660  CALL GENT(ICURSR)
       CALL GPUT(6,73,IX,IY)
       IDX=IX-IXL
       IDY=IY-IYL
       DX=IDX
       DY=IDY
       DS=DX*DX+DY*DY
       CALL GENT(IALT)
       CALL GPUT(IEL,53,IDX,IDY)
       IF(DS.LT.400.)GOTO 510
       JGO=1
C  DRAW VECTOR
  680  IXL=IX
       IYL=IY
       IEL=IEL+1
       IF(NXY.GT.800)GOTO 9000
       IXY(NXY)=IX
       IXY(NXY+1)=IY
       NXY=NXY+2
       GOTO(690,549),JGO
C  CHECK FOR AUTOMATIC CLOSING
  690  IDX=IXP-IXL
       IDY=IYP-IYL
```

```
      DX=IDX
      DY=IDY
      DS=DX*DX+DY*DY
      IF(DS.GE.400.)GOTO 510
      GOTO 552
 750  CONTINUE
      IF(NXY.GT.800)GOTO 9000
      CALL GLMP(1,5,0)
      CALL GLMP(1,25,0)
      CALL GLMP(1,26,0)
C     WRITE END OF ALTITUDE ONTO OUTPUT FILE
      IXY(NXY)=3000
      IXY(NXY+1)=3000
      NXY=NXY+1
 800  CONTINUE
      CALL GEOF(ICURSR)
      CALL GLMP(1,-1,0)
      CALL GEOF(IMAG)
      CALL GCYT(0)
      RETURN
 9000 WRITE(1,1001)
 1001 FORMAT(' ERROR: NO MORE ROOM IN STORAGE VECTOR IXY')
      RETURN
 5020 WRITE(1,1002)
 1002 FORMAT(' ERROR: NOT ENOUGH ROOM ON DISK')
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
C
C
      SUBROUTINE TRACKB(IX,IY)
      COMMON /ATT/IATT(12)
C   FIND TRACKBALL POSITION
C   RESTRICTIONS:
C     CYCLE TIMER ALREADY TURNED ON
C   CAUTION:
C     ALL OTHER INTERRUPTS ARE IGNORED EXCEPT THOSE FROM KEYBOARD,
C     WHICH CAUSE IMMEDIATE RETURN
C     IF KEYBOARD INTERRUPT OCCURRED, IX AND IY ARE UNCHANGED
C
  300 IATT(1)=0
      CALL GSTT(0,0)
  310 CONTINUE
      IF (IATT(1).EQ.0) GO TO 310
      IF (IATT(1).EQ.36) RETURN
      IF (IATT(1).NE.30) GO TO 300
      CALL GDEV(1,IX,IY)
      IX=IX/5+500
      IY=IY/5+500
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE SAMPLE
      COMMON /COMUN/NOVPTR,NNOVLY(20),IFLAG(20)
      COMMON /MIME2/NCNTUR,PROBC(5),NRIPC(5,3),NIXY(5,3),NRPATH,NIPATH,
     1                  IXMIN(2),IXMAX(2),IYMIN(2),IYMAX(2)
      COMMON /SAM/IS(2,100,3),NI(5,3)
      COMMON /ATT/IATT(12)
      COMMON /DISK/IFCB(13)
      DIMENSION IXY(800)
      WRITE(5,1313)
 1313 FORMAT('ENTERED SAMPLE')
      CALL GBEG(20,0,0)
      IELM=6
C
C SET UP NO. OF SAMPLES IN EACH ANNULUS
C
      NCNM1=NCNTUR-1
      ISUM=0
      DO 10 I=1,NCNM1
      NI(I,1)=100.*(PROBC(I)-PROBC(I+1))+0.5
      ISUM=ISUM+NI(I,1)
   10 CONTINUE
      NI(NCNTUR,1)=100.*PROBC(NCNTUR)+0.5
      ISUM=ISUM+NI(NCNTUR,1)
      CALL GHLT
      WRITE(5,2323) NCNTUR,ISUM,(NI(I,1),I=1,NCNTUR)
 2323 FORMAT(12I10)
C
C CHECK AND MODIFY
C
      NI(1,1)=NI(1,1)-ISUM+100
      DO 20 I=1,NCNTUR
      NI(I,2)=NI(I,1)
   20 CONTINUE
C
C SAMPLE
C
      DO 300 IF=1,2
      DXI=IXMAX(IF)-IXMIN(IF)
      XI=IXMIN(IF)
      DYI=IYMAX(IF)-IYMIN(IF)
      YI=IYMIN(IF)
      ISAMP=0
  100 IX=RAN(0)*DXI+XI+0.5
      IY=RAN(0)*DYI+YI+0.5
      DO 200 ICNTUR=1,NCNTUR
      IFCB(4)=NRIPC(ICNTUR,IF)
      NXY=NIXY(ICNTUR,IF)
      READ(18) (IXY(I),I=1,NXY)
C ONLY SIMPLE CONTOURS
      IF(INSIDE(IX,IY,IXY,NXY).EQ.1) GO TO 150
C OUTSIDE CONTOUR ICNTUR
      IF(ICNTUR.EQ.1) GO TO 100
      GO TO 201
  150 IF(ICNTUR.NE.NCNTUR) GO TO 200
C INNERMOST CONTOUR
      ITT=NCNTUR
      GO TO 202
  200 CONTINUE
  201 ITT=ICNTUR-1
C ALREADY ENOUGH SAMPLES???
  202 IF(NI(ITT,IF).EQ.0) GO TO 100
C OK
      NI(ITT,IF)=NI(ITT,IF)-1
      ISAMP=ISAMP+1
      IS(1,ISAMP,IF)=IX
      IS(2,ISAMP,IF)=IY
      CALL DOT(IELM,IX,IY)
      IF(ISAMP.EQ.100) GO TO 300
```

```
PAGE     2              MIMEII     VORTXII    FTN IV(G)              0012 HOURS
              GO TO 100
         300 CONTINUE
              CALL INTRPT
         998 CALL INTRPT
              IF(IATT(1).NE.36) GO TO 998
              IF(IATT(3).NE.0) GO TO 998
              NOVPTR=0
              NMOVLY(1)=4
              RETURN
              END
  0 ERRORS COMPILATION COMPLETE
 /WEOF,14
 /MEM,6
 /FORT,M,H,F
```

```
      SUBROUTINE MIMEII
      COMMON /MIME2/NCNTUR,PROBC(5),NRIPC(5,3),NIXY(5,3),NRPATH,NIPATH,
     1                 IXMIN(2),IXMAX(2),IYMIN(2),IYMAX(2)
      COMMON /CONTUR/NXY,IXY(800),ICOLOR(2),LST,ICURSR,IALT,IM3G
      COMMON /COMUN/NOVPTR,NMOVLY(20),IFLAG(20)
      COMMON /DISK/IFCB(13)
      COMMON /ATT/IATT(12)
      COMMON /SAM/IS(2,100,3),NI(5,?)
      COMMON /MISS/NRS
      EQUIVALENCE (ICNTUR,IFLAG(2))
      IF(IFLAG(1).EQ.1) GO TO 600
      IFLAG(1)=1
      WRITE(5,4040)
 4040 FORMAT(' ENTERED MIME00')
      IFCB(4)=NRPATH
      NXY=NIPATH
      READ(18) (IXY(I),I=1,NXY)
      LPATH=10*(NXY-6)
      IXA0=IXY(1)
      IYA0=IXY(2)
      IXB0=IXY(NXY-5)
      IYB0=IXY(NXY-4)
      WRITE(5,66) NXY,IXA0,IYA0,IXB0,IYB0
   66 FORMAT(10I12)
      IPTCUR=1
      IXCUR=IXY(2*IPTCUR-1)
      IYCUR=IXY(2*IPTCUR)
      CALL GBEG(21,IXCUR,IYCUR)
      CALL GPUT(6,1000,8,0)
      CALL GPUT(7,1000,5,0)
C DRAW CURSOR CIRLCES
   10 CALL INTRPT
      IF(IATT(1).NE.26) GO TO 10
      IF(IATT(3).EQ.31) GO TO 20
      IPTCUR=IPTCUR+1
      IXCUR=IXY(2*IPTCUR-1)
      IYCUR=IXY(2*IPTCUR)
      CALL GPUT(1,100,IXCUR,0)
      CALL GPUT(2,110,IYCUR,0)
      GO TO 10
C COMPUTE ROTATION ANGLES AT EACH POINT
   20 DY=IXY(4)-IXY(2)
      DX=IXY(3)-IXY(1)
      ANGA=ATAN2(DY,DX)
      WRITE(5,1222) DX,DY,ANGA
 1222 FORMAT(3E14.6)
      DY=IXY(NXY-4)-IXY(NXY-6)
      DX=IXY(NXY-5)-IXY(NXY-7)
      ANGB=ATAN2(DY,DX)
      WRITE(5,1222) DX,DY,ANGB
      DY=IXY(2*IPTCUR+2)-IXY(2*IPTCUR-2)
      DX=IXY(2*IPTCUR+1)-IXY(2*IPTCUR-3)
      ANGC=ATAN2(DY,DX)
      WRITE(5,1222) DX,DY,ANGC
      N1=IPTCUR-1
      NT=(NXY-6)/2
      N2=NT-N1
      CALL GBEG(22,0,0)
      IELM=6
      DO 500 I=1,100
      IXA=IS(1,I,1)-IXA0
      IYA=IS(2,I,1)-IYA0
      CALL ROTATE(IXA,IYA,-ANGA)
      IXB=IS(1,I,2)-IXB0
      IYB=IS(2,I,2)-IYB0
      CALL ROTATE(IXB,IYB,-ANGB)
      IXB=IXB+LPATH
      DD=N2*IXA+N1*IXB
```

```
      DD=DD/FLOAT(NT)
      IXC=DD+0.5
      DD=N2*IYA+N1*IYB
      DD=DD/FLOAT(NT)
      IYC=DD+0.5
      IXC=IXC-20*N1
      CALL ROTATE(IXC,IYC,ANGC)
      IXC=IXC+IXCUR
      IYC=IYC+IYCUR
 4566 FORMAT(12I10)
      IS(1,I,3)=IXC
      IS(2,I,3)=IYC
      CALL DOT(IELM,IXC,IYC)
  500 CONTINUE
  998 CALL INTRPT
      IF(IATT(1).NE.30) GO TO 992
      IF(IATT(3).NE.0) GO TO 998
      NOVPTR=0
      NMOVLY(1)=2
      NMOVLY(2)=4
      NMOVLY(3)=-1
      IFCB(4)=NRS
      ICNTUR=0
  550 ICNTUR=ICNTUR+1
      IALT=IALT+1
      RETURN
  600 NIXY(ICNTUR,3)=NXY
      NRIPC(ICNTUR,3)=IFCB(4)
      WRITE(18) (IXY(I),I=1,NXY)
      NI(ICNTUR,3)=0
      DO 700 I=1,100
      IX=IS(1,I,3)
      IY=IS(2,I,3)
      ITEMP=INSIDE(IX,IY,IXY,NXY)
      IF(ITEMP.EQ.1) NI(ICNTUR,3)=NI(ICNTUR,3)+1
  700 CONTINUE
      CALL CHLT
      WRITE(5,999) ICNTUR
      WRITE(5,999) (NI(I,3),I=1,NCNTUR)
      IF(ICNTUR.EQ.NCNTUR) GO TO 800
      GO TO 550
  800 CALL GHLT
  999 FORMAT(50X,5I10)
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
```

```
PAGE    1              MINEII     VORTXII    FTN IV(G)              0013 HOURS
      C
      C
            SUBROUTINE ROTATE(IX,IY,ANG)
            X=IX
            Y=IY
            XT=COS(ANG)*X-SIN(ANG)*Y
            YT=SIN(ANG)*X+COS(ANG)*Y
            IX=XT+0.5
            IY=YT+0.5
            RETURN
            END
 0 ERRORS COMPILATION COMPLETE
```

```
      FUNCTION INSIDE(IX,IY,IXY,NXY)
C INSIDE RESTURNS 0 IF (IX,IY) IS NOT INSIDE THE CURVE SPECIFIED BY THE
C ARRAY IXY. IF THE POINT IS INSIDE THE CURVE, 1 IS RETURNED
      DIMENSION ITBL(4,4),IXY(2)
      DATA ITBL/0,-1,1000,1,1,0,-1,1000,1000,1,0,-1,-1,1000,1,0/
      INSIDE=1
      ICNT=0
      IDX=IXY(1)-IX
      IDY=IXY(2)-IY
      JGO=1
      GO TO 5000
2000  KK=NXY-4
      DO 1000 J=3,KK,2
      IQDL=IQD
      IDX=IXY(J)-IX
      IDY=IXY(J+1)-IY
      JGO=2
      GO TO 5000
3000  IF (ITBL(IQDL,IQD).EQ.1000) GO TO 1100
      ICNT=ICNT+ITBL(IQDL,IQD)
      GO TO 1000
5000  CONTINUE
      IF ((IDX.GE.0).AND.(IDY.GE.0)) IQD=1
      IF ((IDX.LT.0).AND.(IDY.GE.0)) IQD=2
      IF ((IDX.LT.0).AND.(IDY.LT.0)) IQD=3
      IF ((IDX.GE.0).AND.(IDY.LT.0)) IQD=4
      GO TO (2000,3000,4000),JGO
1100  IQDP=IQD
      X=IX
      Y=IY
      IF (IXY(J-1).GT.IXY(J+1))GO TO 7100
      X1=IXY(J-2)
      Y1=IXY(J-1)
      X2=IXY(J)
      Y2=IXY(J+1)
      GO TO 7200
7100  X2=IXY(J-2)
      Y2=IXY(J-1)
      X1=IXY(J)
      Y1=IXY(J+1)
7200  SM=(Y2-Y1)/(X2-X1)
      DX=X-X1
      SY=Y-Y1
      DY=SM*DX
      IF (SY.LE.DY) GO TO 7300
      IDX=X2-X
      IDY=Y1-Y
      JGO=3
      GO TO 5000
7300  IF (SY.EQ.DY) GO TO 7400
      IDX=X1-X
      IDY=Y2-Y
      JGO=3
      GO TO 5000
7400  INSIDE=0
      RETURN
4000  ICNT=ICNT+2*ITBL(IQDL,IQD)
      IQD=IQDP
1000  CONTINUE
      IF (ICNT.EQ.0) INSIDE=0
      RETURN
1200  INSIDE=0
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
/FILE,BO,,BO
/EMAIN
```

```
PAGE     1                              VORTXII   FTN IV(G)                 0018 HOURS
          SUBROUTINE DOT(IELM,IX,IY)
          CALL GPUT(IELM,100,IX,0)
          CALL GPUT(IELM+1,110,IY,0)
          CALL GPUT(IELM+2,1000,3,0)
          IELM=IELM+3
          RETURN
          END
  0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE DOT(IELM,IX,IY)
      CALL GPUT(IELM,192,IX,0)
      CALL GPUT(IELM+1,110,IY,0)
      CALL GPUT(IELM+2,1600,1,0)
      IELM=IELM+3
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE  INTRPT
C
C SUBROUTINE INTRPT WAITS FOR A DISPLAY INTERRUPT
C
      COMMON /ATT/IATT(10)
      IATT(1)=0
      CALL GSTT(0,0)
    1 CONTINUE
      IF(IATT(1).EQ.0) GO TO 1
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
/PFILE,BO,,BO
/FMAIN
```

```
C
C
      SUBROUTINE SETUPW(IENT,IELM,IX,IY,IWL,ISIZ)
C
C SETUPW SETS UP A CHARACTER DISPLAY AREA OF LENGTH IWL AT (IX,IY)
C WITH CHARACTER SIZE ISIZ BEGINNING AT ELEMENT IELM IN ENTITY IENT,
C AND RETURNS WITH IELM INCREMENTED TO THE NEXT FREE ELEMENT.
      CALL GPUT(IELM,100,IX,0)
      CALL GPUT(IELM+1,110,IY,0)
      CALL GCHA(IENT,IELM+2,0,ISIZ,IWL)
      IELM=IELM+3+IWL
      RETURN
      END
```

0 ERRORS COMPILATION COMPLETE

```
      SUBROUTINE SETUPR(IENT,IELM,IWL,ISIZ)
C
C SETUPR SETS UP A CHARACTER DISPLAY AREA OF LENGTH IWL AND SIZE
C ISIZ BEGINNING AT ELEMENT IELM IN ENTITY IENT, CALLS TEXT FOR
C THE OPERATOR TO ENTER INTO THIS AREA, AND RETURNS WITH IELM
C INCREMENTED TO THE NEXT FREE ELEMENT.
C
      CALL GCHA(IENT,IELM,0,ISIZ,IWL)
      CALL TEXT
      IELM=IELM+1+IWL
      RETURN
      END
0 ERRORS COMPILATION COMPLETE
```

```
      SUBROUTINE TEXT
      COMMON /ATT/IATT(10)
      CALL GCVT(0)
      IATT(1)=0
      CALL GSTT(0,0)
      KURSOR=1
    1 CALL GTEP(KURSOR,FARTRM)
      IF(FARTRM.EQ.-29280) GO TO 2
      IF(IATT(1).EQ.0) GO TO 1
      IATT(1)=0
      CALL GSTT(0,0)
      GO TO 1
    2 CALL GHLT
      RETURN
      END
 0 ERRORS COMPILATION COMPLETE
/FILE,BO,,BO
/EMAIN
```

TECHNOMETRICS©, VOL. 19, NO. 2, MAY 1977

Variable Kernel Estimates of Multivariate Densities

Leo Breiman, William Meisel, and Edward Purcell

Technology Service Corporation
2811 Wilshire Boulevard
Santa Monica, California 90403

A class of density estimates using a superposition of kernels where the kernel parameter can depend on the nearest neighbor distances is studied by the use of simulated data. Their performance using several measures of error is superior to that of the usual Parzen estimators. A tentative solution is given to the problem of calibrating the kernel peakedness when faced with a finite sample set.

KEY WORDS

Density estimation
Kernel density estimation

1. INTRODUCTION AND SUMMARY

Given points $\mathbf{x}_1, \cdots, \mathbf{x}_n$ selected independently from some unknown underlying density $f(\mathbf{x})$ in M-dimensional Euclidean space, the problem is to estimate $f(\mathbf{x})$. To date, the most effective general method is the Parzen approach: select a kernel function $k(\mathbf{x}) \geq 0$, with

$$\int k(\mathbf{x})\, d\mathbf{x} = 1 \tag{1}$$

Usually $k(\mathbf{x})$ satisfies some additional conditions; unimodality with peak at $\mathbf{x} = 0$, smoothness, symmetry, finite first and second moments, etc. In fact, in actual practice, the most frequently used kernel is a Gaussian density.

Having selected a kernel, then the estimate is given as

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{1}^{n} \frac{1}{\sigma^M} K\left(\frac{\mathbf{x} - \mathbf{x}_j}{\sigma}\right)$$

As n increases the shape factor σ can be decreased giving greater resolution for larger sample sizes. The asymptotic mean square consistency of these estimates is well known [7], and under smoothness conditions on $f(\mathbf{x})$ asymptotic rates of convergence of the mean squared error can be derived.

However, in terms of practicalities, the situation is far from satisfactory.

First: It is obvious that a Parzen method of estimation cannot respond appropriately to variations in the magnitude of $f(\mathbf{x})$. For instance, if there is a region of low $f(\mathbf{x})$ containing, say, only one sample point $\mathbf{x}_k$, then the estimate will have a peak at $\mathbf{x} = \mathbf{x}_k$ and be too low over the rest of the region. In regions where $f(\mathbf{x})$ is large, the sample points are more densely packed together, and the Parzen estimate will tend to spread out the high density region. Thus, the problem is that the peakedness of the kernel is not data-responsive.

Second: None of the asymptotic results give any generally helpful leads on how the shape factor σ should be selected to give the "best" estimate of unknown density. The computed rates of convergence depend critically on $f(\mathbf{x})$ and its derivatives. Even if one tried to vary σ and got a number of different estimates, the question remains: which one is "best"?

In this paper, solutions are proposed to both of these problems.

First: To make the sharpness of the kernel data-responsive, we use the class of estimates

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{j=1}^{n} (\alpha_k d_{j,k})^{-M} K\left(\frac{\mathbf{x} - \mathbf{x}_j}{\alpha_k d_{j,k}}\right)$$

where $d_{j,k}$ is the distance from the point $\mathbf{x}_j$ to its kth nearest neighbor, and α_k is a constant multiplicative factor. The intuitive concept is clear: In low density regions, $d_{j,k}$ will be large and the kernel will be spread out. In high density regions, the converse will occur.

Second: To select optimizing values of k and α_k, a goodness-of-fit statistic $\mathcal{S}$ for multivariate densities proposed in [1] is used in a procedure that searches for the variable kernel parameters that minimize $\mathcal{S}$, (see section 3 for the definition of $\mathcal{S}$).

Research sponsored by the *Air Force Office of Scientific Research, AFSC, United States Air Force*, under Contract No. F44620-71-C-0093. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Received July 197[illegible], revised May 1976

There is a large body of published literature regarding density estimation and a number of good surveys are available [4], [9], [2]. The kth nearest neighbor estimator [5] is the only method that is adaptive to local sample density. If the distance from a point x to its kth nearest neighbor is d, then this estimator is defined as

$$\hat{f}(\mathbf{x}) = \frac{k/n}{V(d)}$$

where n is the total number of samples, and $V(d)$ is the volume of the M-dimensional sphere of radius d. The drawback to this type of estimate is that it is discontinuous. (Also its integral over all space is infinite.) The variable kernel approach offers a combination of the desirable smoothness properties of the Parzen-type estimators with the data-adaptive character of the k-nearest neighbor approach.

Furthermore, the variable kernel method carries very little computational penalty. The distance from a given point to the kth nearest point is computed only once and stored for all the calibration runs. An algorithm constructed by Friedman, et al. [3] reduces the finding of all kth nearest neighbors to $n \log n$ time instead of n^2.

The analytics of the situation are a bit difficult to handle, although asymptotic consistency for appropriate kernels is easily proved under the condition $k/n \to 0$. To get a feeling for the finite sample situation and also to get some measure of assurance that our proposed "solutions" had some value, we ran some extensive simulations on two underlying data bases; the first was 400 points selected from a bivariate normal distribution, the second was 400 points selected from a bimodal distribution consisting of a superposition of two bivariate normals, 3/4 of the bivariate normal used in generating the first data set plus 1/4 of a normal with a much sharper peak.

Three measures of error were computed: define the sample mean and variance of $f(\mathbf{x})$ by

$$\hat{\mu}_f = \frac{1}{n} \sum_{1}^{n} f(\mathbf{x}_j)$$

$$\hat{\sigma}_f^2 = \frac{1}{n} \sum_{1}^{n} (f(\mathbf{x}_j) - \hat{\mu}_f)^2$$

The error measures were:

(PVNE) Percent of Variance Not Explained

$$\mathrm{PVNE} = \frac{1}{\hat{\sigma}_f^2} \cdot \frac{1}{n} \sum_{1}^{n} (f(\mathbf{x}_j) - \hat{f}(\mathbf{x}_j))^2 \times 100$$

(MAE) Mean Absolute Error, Percent

$$\mathrm{MAE} = \frac{1}{n\hat{\mu}_f} \sum_{1}^{n} |f(\mathbf{x}_j) - \hat{f}(\mathbf{x}_j)| \times 100$$

(MPE) Mean Percent Error

$$\mathrm{MPE} = \frac{1}{n} \sum_{1}^{n} \frac{|f(\mathbf{x}_j) - \hat{f}(\mathbf{x}_j)|}{f(\mathbf{x}_j)} \times 100$$

A large number of runs were carried out with the two data bases to

- For each measure of error find the "best" Parzen estimator and the "best" variable kernel estimator, using a symmetric Gaussian kernel (i.e., find those values of the parameters σ, k, α_k that minimize the given measure of error).
- Compare the performances of the two types of estimators.
- To see whether the proposed search procedure could accurately locate the "best fitting" parameter values.

Our conclusions are:

i. In all cases the best variable kernel estimator was superior to the best Parzen estimate. The best Parzen estimator, in both data sets, had about twice as much mean percent error (MPE) and percent of variance not explained (PVNE), and about 50% more mean absolute error than the best variable kernel estimator.

ii. The $\hat{S}$ minimization search procedure was successful in locating the region of parameter values where the variable kernel estimates gave approximately best fits to the actual density.

The best values of σ for the Parzen estimates depended on which measure of error was used much more than the variable kernel method and hence would be much more difficult to use in practice (when f is unknown). The $\hat{S}$ minimization procedure applied to the Parzen estimates produced values of σ that were larger than most of the "best" values and could not be called successful in this context.

During the course of the study, a number of interesting and useful properties of variable kernel densities were uncovered. Recalling that the total sample size is 400, nearest neighbor distances that produced the best fits were surprisingly large, ranging from 40 in data set II to 100 in data set I, (actually the fit was still improving at $k = 100$). But good fits can be produced over a very wide range of values of k, as long as α_k satisfies the approximate relation

$$\frac{\alpha_k(\bar{d}_k)^2}{\sigma(d_k)} = \text{constant}$$

where $\bar{d}_k$ is the mean of the kth nearest neighbor distances and $\sigma(d_k)$ is their standard deviation. Our tentative conclusion is therefore that actually one needs to find only the single parameter value $[\alpha_k(\bar{d}_k)^2/\sigma(d_k)]$ to calibrate the variable kernel estimates. In our simulation this constant was usually about 3-4 times larger than the best values of σ for the corresponding Parzen estimate.

The conclusion that the mean percent error is markedly different between the two types of estimators has important implications for classification. The method giving the minimum expected misclassification probability is based on comparing the densities of the different classes. One common and effective method of getting "good" classification boundaries has been to estimate the class densities using a set of points that have already been classified, and to compare the density estimates to make the classification decision. If this is the intended application, then the mean percent error may be the most significant error measure, since it is the ratio of the two estimates that determines the classification. In this perspective the variable kernel estimates are decidedly superior to the Parzen estimates.

An important consideration is the variability of the underlying density. If it is more or less uniformly smooth (as in the first data base), the adaptive capability of the variable kernel method does not help us much as in situations where the density is more variable, i.e., has a number of peaks of different sharpness (as in the second data base).

The body of the paper is laid out as follows: Section 2 describes the simulations in more detail and includes some tabular and graphical summaries of the results. Section 3 contains a brief description of the goodness-of-fit statistic together with tabular and graphical summaries of its performance. Section 4 summarizes the behavior of the estimates, relates the selection of k and α to the interpoint distance distribution, and gives a description of some early and unsuccessful efforts at variable kernel estimates.

The variable kernel method has been described in short course notes on pattern recognition prepared by one of the authors and dating back to 1973. Part of the work in this present study was presented in the Conference on the Interface Between Computer Science and Statistics on February 14, 1975 [6]. In June, 1975 we learned that T. J. Wagner had submitted a paper to the IEEE Trans. Information Theory which is also concerned with the variable kernel estimates. His paper [8] establishes conditions for asymptotic consistency, particularly in one dimension.

2. THE SIMULATION AND ITS RESULTS

The two data sets mentioned in the introduction were generated as follows:

Set I: 400 points selected independently from the density f, a bivariate normal with mean $m = (0, 0)$ and unit covariance matrix.

Set II: 400 points selected independently from the density g, where

$$g = .75f + .25f_1$$

where f is as above, and f_1 is normal with parameters

$$m = (3, 3), \qquad \Gamma = \begin{pmatrix} 1/9 & 0 \\ 0 & 1/9 \end{pmatrix},$$

where Γ is the covariance matrix.

The kernel for both types of estimators was a zero mean bivariate normal density with unit covariance matrix.

FIGURE 1 is a graph of the three error measures in data set I as a function of the shape parameter σ of the Parzen estimators.

FIGURE 2 is a graph of the three error measures for data set I, where we selected $k = 100$ and varied the multiplicative parameter α.

FIGURES 3 and 4 are the analogous graphs for

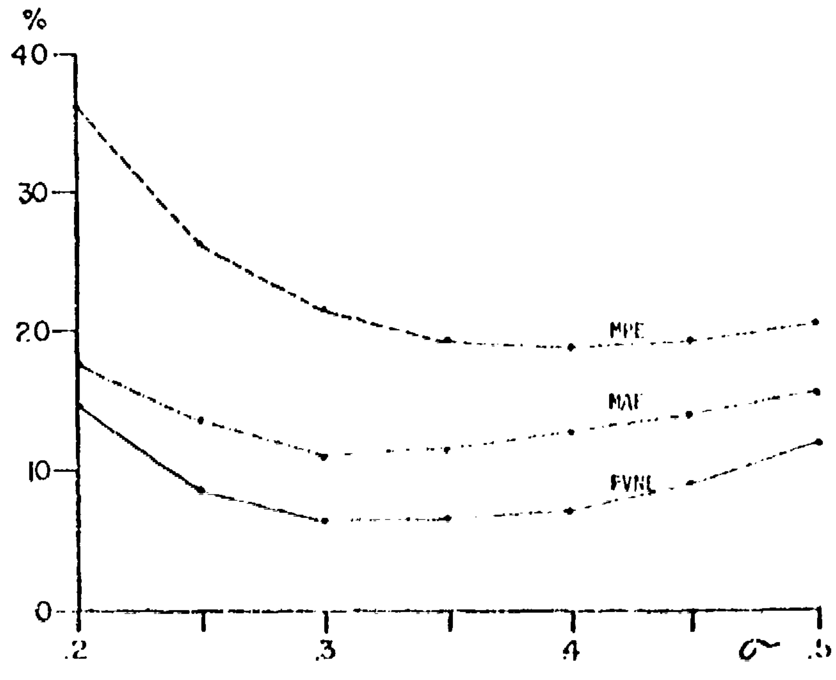


FIGURE 1—Measures of Error for the Parzen Estimator, Data Set I

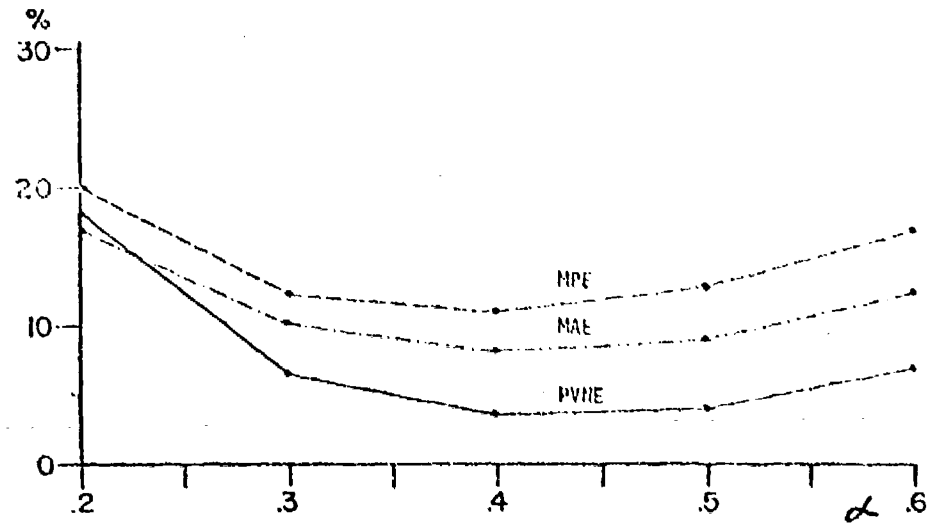


FIGURE 2—Measures of Error for the Variable Kernel Estimator, $k = 100$, Data Set I

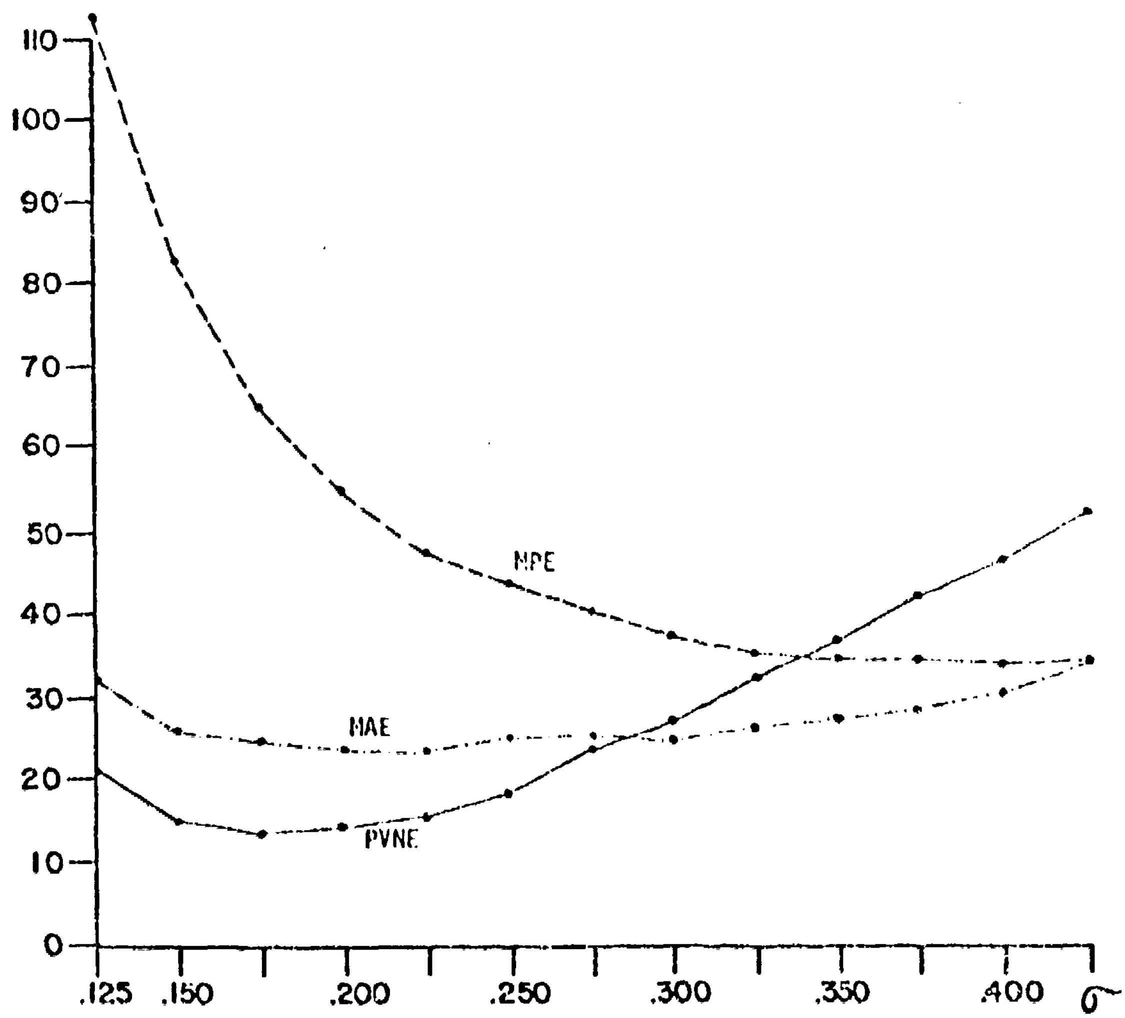


FIGURE 3—Measures of Error for the Parzen Estimator, Data Set II

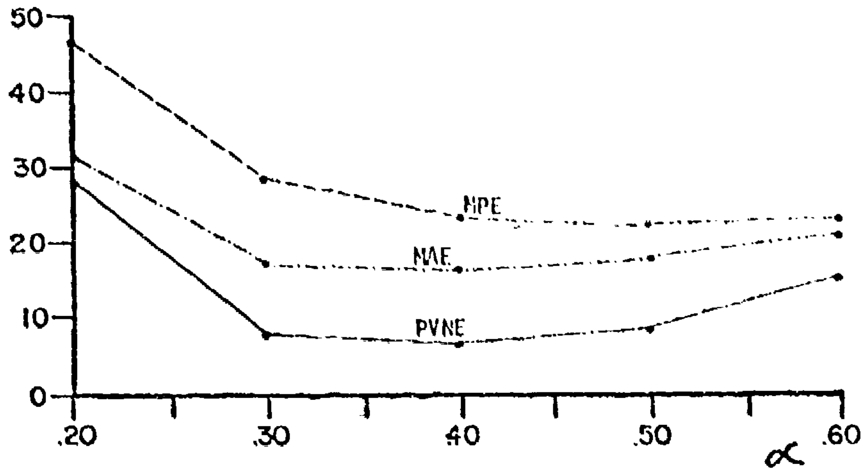


FIGURE 4—Measures of Error for the Variable Kernel Estimator, $k = 40$, Data Set II

data set II, where we have used $k = 40$ in the variable kernel graph.

In all cases, we ran the simulations until the minimal values of the three measures of error were found, both for the Parzen and variable kernel estimators. For the variable kernel estimators we ran the simulations for $k = 10, 20, 30, 40, 50$, and 60 in both data sets, and for $k = 70, 80, 90, 100$ in data set I. Table 1 below summarizes the comparison between the methods.

To illustrate the resulting fits more visually, we plotted 3 dimensional graphs of the best estimates. For data set I, we used $\sigma = .35$ for the Parzen estimator and $k = 60$, $\alpha = .6$ for the variable kernel estimator. In data set 2, the choice of an "optimal" σ was more problematical. We settled on .275 as a reasonable compromise. For the variable kernel we took $k = 40$, $\alpha = .5$. The results are shown in figures 5, 6, 7, and 8 (see end).

Fortunately, the variable kernel results were surprisingly insensitive to the choice of k. Table 2 below gives the minimum values of the measures of error for the different values of k. Note that in both examples, values of k over almost the entire range give quite comparable error measurements.

As k varies the fit behaves slightly differently for the two data sets. For the smooth density of the first example, the error measures are still decreasing at $k = 100$ and we would probably have gotten slightly better results by going on to larger k. For the second density the error measures decrease up to $k = 40$ and then increase at $k = 50$ and 60, (except for the MPE).

While the best fit for each value of k in a wide range has about the same error measures, the values of the multiplier α at which the minimum errors occur vary considerably but systematically as k increases. We will explore this further in Section 4.

3. THE GOODNESS-OF-FIT CRITERION

Since, in practice, the underlying $f(x)$ is not known, the various error measures cannot be computed. This brings us to the second question posed in the introduction: How then do we go about selecting σ or α_k and k. (Although we surmise that in actuality we

TABLE 1—*Comparison Between Methods*

	Minimum Mean Percent Error	Minimum Percent of Variance Not Explained	Minimum Mean Absolute Error, Percent
Parzen, Data Set I	19.0	6.2	11.6
Variable Kernel, Data Set I	10.8	3.6	8.0
Parzen, Data Set II	34.7	13.4	24.2
Variable Kernel, Data Set II	27.5	6.2	16.5

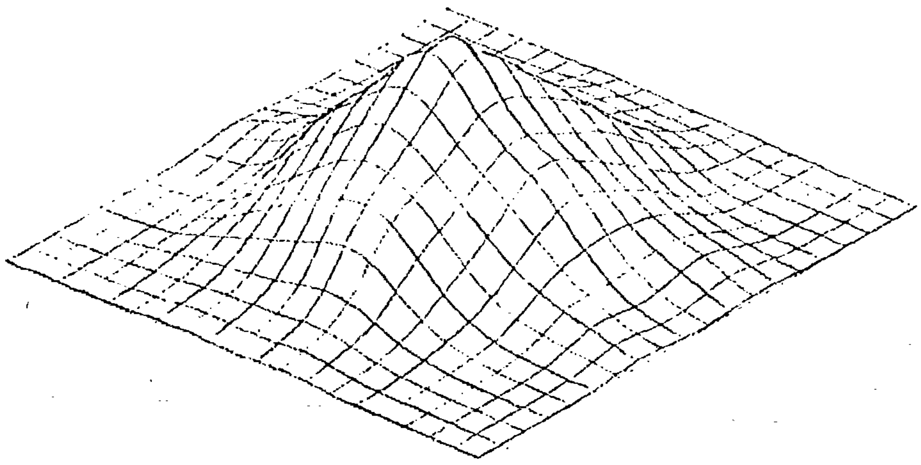


FIGURE 5—*Constant* kernel fit to UNIT NORMAL
$\sigma = .35$

need to estimate only the optimal single parameter value $\lambda = \alpha_k(\overline{d_k})^2/\sigma(d_k)$ in the variable kernel estimates.)

In [2] a goodness-of-fit criterion for a set of samples to a proposed density $f(\mathbf{x})$ was developed based on the fact that if $f(\mathbf{x})$ is the true density, then the variables

$$w_j = e^{-nf(\mathbf{x}_j)V(d_{j,k})}, \qquad j = 1, \cdots, n$$

where $V(r)$ is the volume of an M-dimensional sphere of radius r, ($V(r) = \Pi^{M/2}r^M/\Gamma(M/2 + 1)$), have a univariate distribution that is approximately uniform. Thus, the test statistic for an estimate $\hat{f}(\mathbf{x})$ is based on the variables

$$\hat{w}_j = e^{-n\hat{f}(\mathbf{x}_j)V(d_{j,k})}, \qquad j = 1, \cdots, n.$$

Let $\hat{w}_{(1)} \leq \cdots \leq \hat{w}_{(n)}$ be the ordered permutation of the $\hat{w}_j$. Then the test statistic $\hat{S}$ is defined as

$$\hat{S} = \sum_{j}^{n} \left(\hat{w}_{(j)} - \frac{j}{n}\right)^2.$$

One question of great interest to us in this study was whether we could select "good" values of σ or k and α_k searching for a minimum in $\hat{S}$. The results were affirmative (with one exception we will discuss later). Naturally, different error measures were generally minimized at different values of the parameters. In Table 3 we list, for every value of k used, the value of α that minimizes each error measure and the value of α that minimizes $\hat{S}$ for that value of k.

For the unimodal case the absolute minimum of $\hat{S}$ occurs at $k = 100$, $\alpha = 5$. At this point we have

Mean Percent Error = 12.5 (10.8)
Percent of Variance Unexplained
= 4.2 (3.6)
Mean Absolute Error, Percent = 8.8 (8.0).

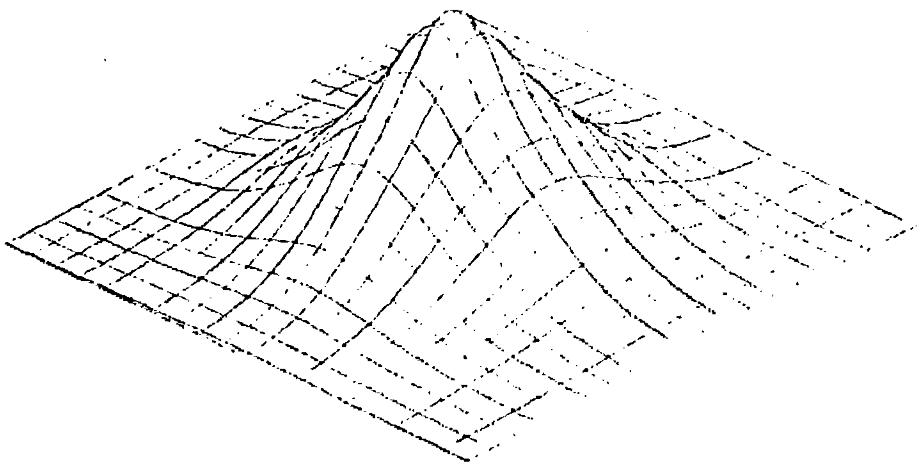


FIGURE 6—*Variable* kernel fit to UNIT NORMAL
$k = 60 \quad \lambda = .6$

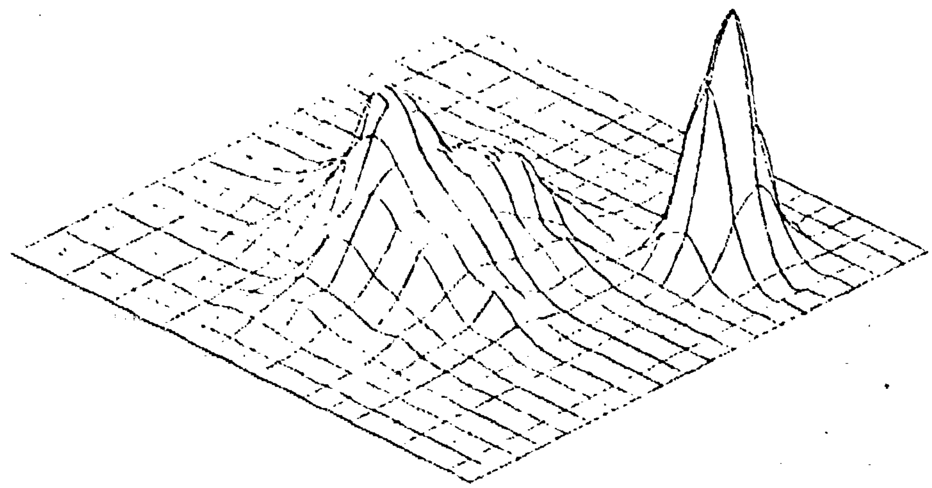


FIGURE 7--*Constant* kernel fit to BIMODAL
$\sigma = .275$

The figures in parentheses are the minimums of the corresponding measures of error over all ranges and do not occur at a common value of k and α.

In the bimodal case, the minimum of $\hat{S}$ occurred in the original runs at $k = 60$, $\alpha = .4$. The values at this point were farily close to the minimums, i.e.,

Mean Percent Error = 22.8 (22.5)
Percent of Variance Unexplained
= 10.7 (6.2)
Mean Absolute Error, Percent = 18.8 (16.5).

For the Parzen Estimator with data set I, the minimizing values of σ for the three error measures above were .40, .35 and .30 respectively. The minimum value of $\hat{S}$ occurred at .60. For data set II, the minimums occurred at .400, .175, .225 and the minimum of $\hat{S}$ at .375. For Parzen estimators $\hat{S}$ indicates "optimal" values of σ considerably higher than the values of σ that minimize the PVNE and the MAE. There is also less consistency between the error measures as to the location of the respective minimizing σ. The σ that minimizes the mean percent error is the highest, and in the bivariate case, considerably higher than the other two minimizing values of σ. Probably this latter fact is due to the behavior of the Parzen estimates at small values of $f(x)$.

In both data sets, the $\hat{S}$ estimate of σ gives a value of mean percent error close to the minimum attain-

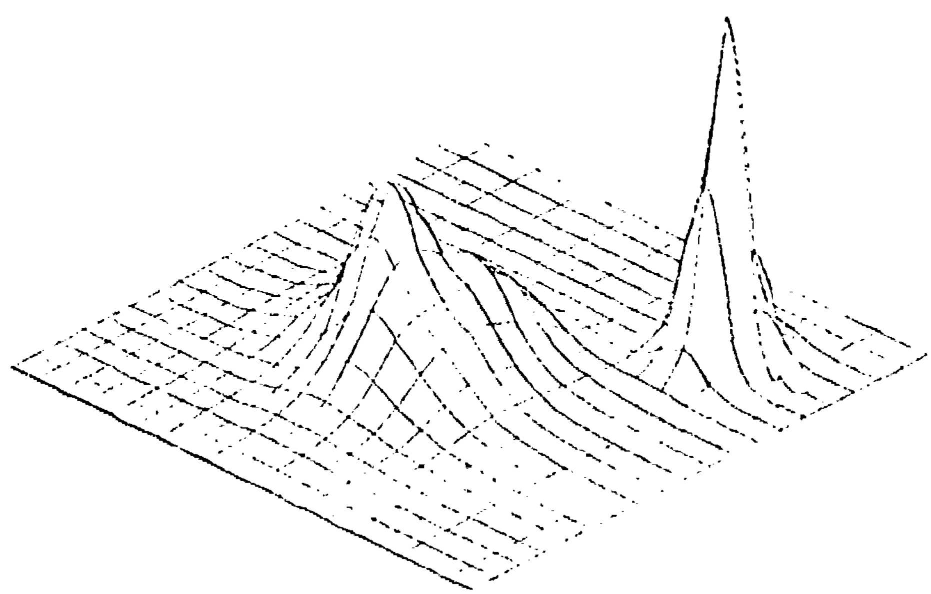


FIGURE 8--*Variable* kernel fit to BIMODAL
$k = 40 \quad \alpha = .5$

TABLE 2—*Minimum values of the measures of error for the different values of k.*

Data Set I

k =	10	20	30	40	50	60	70	80	90	100
Minimum Mean Percent Error	12.9	12.9	12.2	12.1	11.7	11.6	11.4	11.3	10.9	10.8
Minimum Percent of Variance Not Explained	9.3	6.8	6.3	5.9	5.1	4.8	4.6	4.1	4.0	3.6
Minimum Mean Absolute Error, Percent	11.7	11.2	10.7	10.3	9.7	9.3	9.3	8.6	8.5	8.5

Data Set II

k =	10	20	30	40	50	60
Minimum Mean Percent Error	24.5	23.8	23.0	22.6	22.5	22.8
Minimum Percent of Variance Not Explained	9.4	7.6	6.8	6.2	6.4	6.5
Minimum Mean Absolute Error, Percent	19.1	17.9	17.1	16.5	17.2	16.9

able for the data set. This is consistently true for the variable kernel estimates also. For each value of k, the $\hat{S}$ minimizing value of α_k has a mean percent error close to the minimum possible for that value of k.

4. MEAN INTERPOINT DISTANCE AND THE CHOICE OF α

In our various explorations of the variable kernel estimates, we made the empirical discovery that for both data sets, over the range of k investigated,

$$\frac{\alpha_k(\overline{d_k})^2}{\sigma(d_k)} \simeq \text{constant}$$

where $\overline{d_k}$ and $\sigma(d_k)$ are the mean and standard deviation of the kth nearest neighbor distances for the data set, and α_k is the "optimal" α for that value of k. To illustrate this, we use as the "optimal" value of α_k, the average of the first three minimizing values given in Table 3. Table 4 gives the values of $\alpha_k(\overline{d_k})^2/\sigma(d_k)$.

The constant decreases about 40% between the two data sets. A similar decrease occurs for those values of σ in the Parzen Estimates which minimize the Mean Absolute Error % and the Percent of Variance Not Explained. It seems clear that the increase in optimal kernal sharpness occurs in order to deal with the increased variability in data set #2.

At the beginning of this study, we used distances to the closest neighbor, next closest neighbor, etc., up to the fifth nearest neighbor. The results were disastrous. Examining the errors, they came mainly from a few points that were too close together. We tried a number of things:

i. Selecting a lower bound D for the interpoint distances and using

$$d'_{j,k} = \max(D, d_{j,k})$$

in the kernel estimate of $d_{j,k}$. D was selected as a percentile (usually either the 5th or 10th) of the $d_{j,k}$, $j = 1, \cdots, 400$.

ii. Using a weighted average of the first k nearest neighbor distances.

iii. Selecting a multiplicative constant α_k and using $\alpha_k d_{j,k}$ or $\alpha_k d'_{j,k}$.

None of these helped very much as long as we kept working with k small. The averaging in (ii) was no help. Later we made a theoretical computation in order to find values $\alpha_1, \cdots, \alpha_k$ with

$$\alpha_i \geq 0, \qquad i = 1, \ldots, k, \qquad \sum_1^k \alpha_i = 1$$

and such that the variance of

$$\sum_1^k \alpha_i d_{j,i}$$

is a minimum. Assuming that the density was "locally constant" so that the distribution of points is "locally Poisson," the answer is

$$\alpha_1 = \alpha_2 = \cdots = \alpha_{k-1} = 0, \qquad \alpha_k = 1$$

This result gave us some insight into the failure of the averaging process.

TABLE 3—*Minimizing Values of* α_k.

DATA SET I

k =	10	20	30	40	50	60	70	80	90	100
Mean Percent Error	1.4	1.0	0.8	0.7	0.6	0.5	0.5	0.4	0.4	0.4
Percent of Variance Unexplained	1.8	1.2	1.0	0.8	0.7	0.6	0.5	0.5	0.5	0.4
Mean Absolute Error, Percent	1.5 or 1.6	1.0	0.9	0.7	0.7	0.6	0.5	0.5	0.4	0.4
$\hat{S}$	1.7	1.2	0.9	0.8	0.7	0.7	0.6	0.6	0.5	0.5

DATA SET II

k =	10	20	30	40	50	60
Mean Percent Error	1.4	0.9	0.7	0.6	0.5	0.4
Percent of Variance Unexplained	1.0	0.6	0.5	0.4	0.3	0.3
Mean Absolute Error, Percent	1.0	0.6	0.5	0.4	0.3	0.3
$\hat{S}$	1.1	0.8	0.6	0.5	0.5	0.4

In (iii) we found that trying to get more smoothing by increasing α_k led to serious underestimates of the peaks of the densities.

Nothing really helped until we started exploring the larger values of k and found that (iii) worked well when k was large enough.

In terms of what has been empirically learned in this study, we tentatively propose the following method for calibrating a variable kernel density estimate.

Step 1. Pick an initial k equal to some fraction of the sample size, say 10%, or by plotting $\overline{d_k}$ versus k and taking a value of k past the knee of the curve (see figure 9).

Step 2. Do a search for the value of α_k that minimizes $\hat{S}$.

Step 3. Using the minimizing value compute

$$\lambda = \frac{\alpha_k(\overline{d_k})^2}{\sigma(d_k)}$$

Step 4. Vary k in both directions, selecting α_k so as to hold the above ratio constant and search for a k value that minimizes $\hat{S}$.

Note that Step 3 may be dimension dependent.

TABLE 4—$\alpha_k(\overline{d_k})^2/\sigma(d_k)$

k =	10	20	30	40	50	60	70	80	90	100
Data Set #1	1.3	1.3	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Data Set #2	.83	.80	.84	.85	.79	.84	-	-	-	-

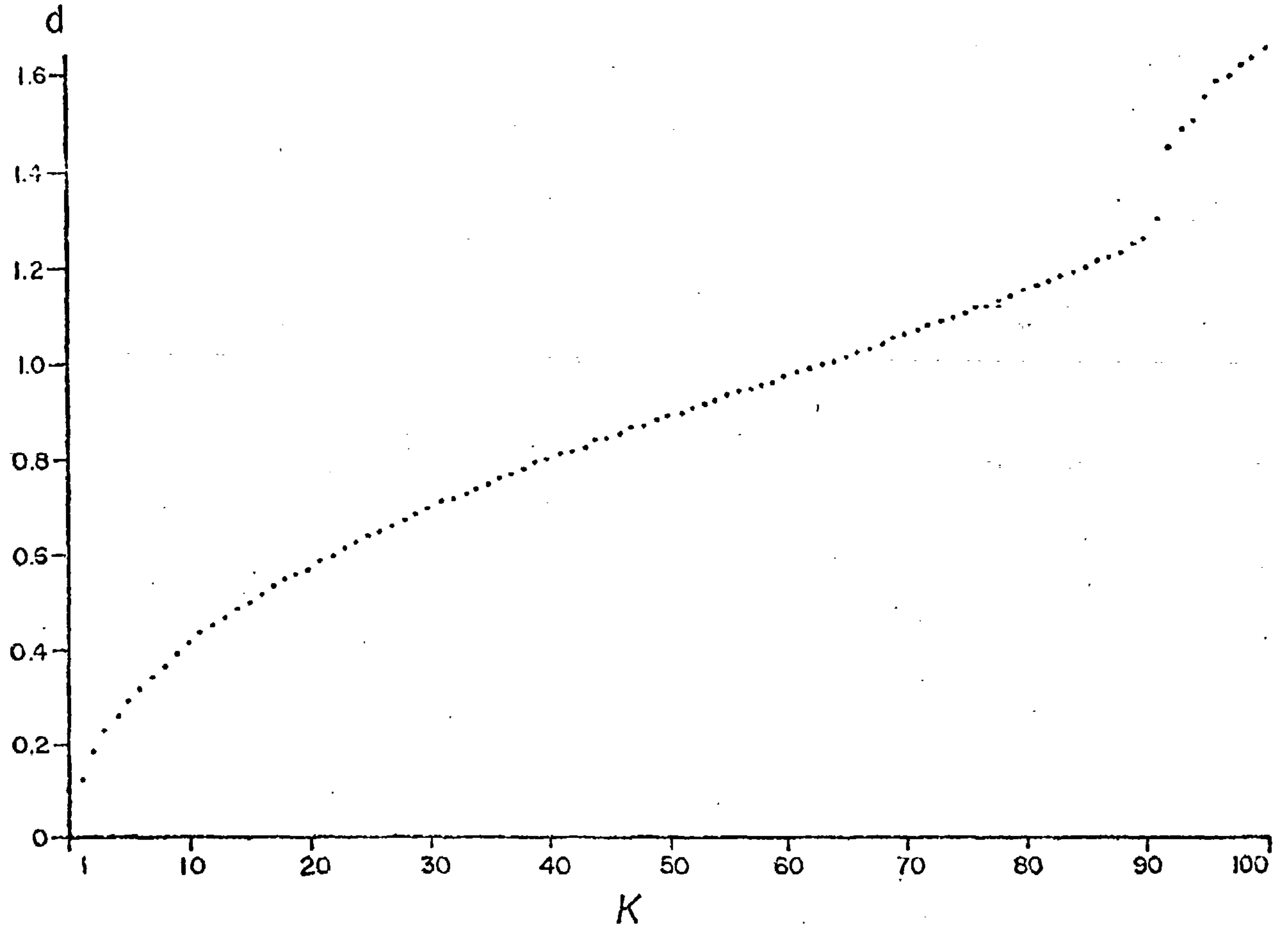


FIGURE 9—The Mean Interpoint Distance $\bar{d}_k$ versus k

REFERENCES

[1] BREIMAN, L. (1975). "A General Goodness-of-fit Test for Multidimensional Densities", submitted to *J. Amer. Statist. Assoc.*

[2] COVER, T. M. (1972). "A Hierarchy of Probability Density Function Estimates", *Frontiers of Pattern Recognition*, Academic Press.

[3] FRIEDMAN, J. H., BENTLEY, J. L., and FINKEL, R. A. (1974). "An Algorithm for Finding Best Matches in Logarithmic Time", submitted to *Communications of the ACM.*

[4] KANAL, L. (1974). "Patterns in Pattern Recognition", *IEEE Trans. on Information Theory*, *IT-20*, *6*, 697–722.

[5] LOOFTSGAARDEN, P. O. and QUESENBERRY, C. P. (1965). "A Nonparametric Estimate of a Multivariate Probability Density Function", *Ann. Math. Statist.*, *28*, 1049–1051.

[6] MEISEL, W. (1975). "The Complete Pattern Recognition Algorithm", 8th Annual Symposium on the Interface Between Computer Science and Statistics, Health Sciences Computing Facility, UCLA.

[7] PARZEN, E. (1962). "On the Estimation of a Probability Density Function and the Mode", *Ann. Math. Statist.*, *33*, 1065–1076.

[8] WAGNER, T. J. (1975). "Nonparametric Estimates of Probability Densities", *IEEE Trans. Information Theory*, *IT-21*, *4*.

[9] WEGMAN, E. J. (1972). "Nonparametric Probability Density Estimation I. A Summary of Available Methods", *Technometrics*, *11*, 533–546.

APPENDIX D -- SKETCH MODEL RESEARCH

Integrated Sciences Corporation has developed and researched a military decision aid called the "Sketch Model" procedure. This procedure allows a human operator to communicate to a computer his subjective estimate of the form of any functional relationship that is continuous in at least one dimension. This communication is performed by the operator using an input device to electronically "sketch" the function on a computer graphics display.

The Sketch Model approach to subjective estimation is hypothesized to have certain advantages over comparable decision aiding techniques, such as the scalar-value representation of subjective judgments. These advantages include comprehensiveness of estimation; ease, speed, and accuracy of estimation and of updating; the ability to consider qualitative data; and the capability of allowing Bayesian estimation without explicit <u>a priori</u> data.

A number of these hypothesized advantages have been experimentally validated by research carried out at ISC's Simulation and Display Facility (SDF). This multi-year research program has consisted of four phases:

1. Evaluation of the Ability of Humans to Use the Sketch Model Technique to Estimate a Bivariate Gaussian Density Function from Sampled Data.

2. Evaluation of the Usefulness of Three Control Devices in Generating Contours Required for Sketch Models.

3. Evaluation of the Ability of Humans to Use the Sketch Model Technique to Approximate a Multi-Modal Tactical Function Based on Qualitative and Quantitative Information.

4. Evaluation of the Ability to Use the Sketch Models Produced by Humans to Drive an Operator Aided Optimization Procedure for Tactical Decision Making.

The Phase I project had a variety of purposes but was primarily aimed at determining the accuracy of humans' attempts to use the Sketch Model procedure to estimate a Bivariate Gaussian Density Function (BGDF). A baseline comparison was provided by carrying out a Maximum Likelihood Estimation (MLE) procedure on the same experimental stimuli (data points sampled from trial BGDF's).

An experiment was conducted with the following independent variables incorporated into the design:

- Procedure (MLE vs. human operated Sketch Model)
- Characteristics of true BGDF ($\overline{x}$, $\overline{y}$, σ_x, σ_y, ρ)
- Sample size of data points used by procedure (5, 10, 100)
- Subjects (7 UCLA undergraduates)

The primary dependent variable was error from the true parameter value of the BGDF. The subjects were trained for 40 hours spread over four weeks. The experimental procedure consisted of presenting a subject with a number (5, 10, or 100) of data points sampled from the true BGDF via a graphics CRT terminal. The subject was then asked to develop his estimate of the true BGDF via the Sketch Model procedure.

Using the Sketch Model procedure to develop a model of BGDF consisted of two steps. In the first step, the subject used a light pen to electronically "draw" on the CRT a family (five) of concentric symmetric ellipses superimposed on the sampled data points. These ellipses represented the subject's estimate of a family of iso-probability contours of the true BGDF. The second step consisted of the subject using scalar judgment to estimate the probability represented by each of his drawn contours. He was provided with a histogram feedback of all five probability values. The five contours plus the five probability values completed the Sketch Model estimate of the BGDF.

An adaptation of Green's Theorem was used to operate on the subjects' contours to measure the mathematical moments of the Sketch Model. These moments were then used to determine the Sketch Model's estimates of parameters ($\overline{x}$, $\overline{y}$, σ_x, σ_y, ρ) of the BGDF. The MLE procedure was applied to the same sets of sampled data points shown to the subjects to develop competing estimates of the BGDF parameters. The error histograms were developed for each of the five BGDF parameters for both Sketch Model and MLE procedures. These histograms were compared to investigate which procedure was superior (as measured by probability of a given size of estimation error) and, if so, under what conditions.

Visual inspection and non-parametric statistical tests lead to the following conclusions:

1. With 100 sample points, MLE produced superior estimation performances over the Sketch Model approach.

2. With sample sizes of 10, the two procedures were insignificantly different with respect to estimation performances.

3. With sample sizes of 5, the Sketch Model procedure was superior to MLE.

4. The first three conclusions were true for all five parameters but differed in degree from parameter to parameter, for example:

   (a) The Sketch Model approach was generally better at estimating horizontal and vertical means ($\overline{x}$, $\overline{y}$) because of the ability of the humans to nonlinearly weight the influence of outliers.

   (b) The MLE approach was generally better at estimating the horizontal and vertical standard deviations (σ_x, σ_y).

   (c) The Sketch Model procedure was vastly superior at estimating the correlation coefficient (ρ).

This first research project established beyond any doubt that the Sketch Model procedure was a viable alternative to any military estimation requirement. This is particularly so since the first experiment was a straightforward estimation task without any opportunity for the human subjects to take advantage of qualitative information sources. Wherever these are available, they will likely improve the competitive edge of the Sketch Model approach with respect to completely automatic techniques.

The Phase II research was aimed at evaluating the relative usefulness of three control devices for implementing the Sketch Model procedure. One way to implement a Sketch Model is for the decision maker to "draw" the curve, line, or surface representing his estimate on a computer-driven graphics display. Various types of control hardware are available to perform drawing tasks in conjunction with an interactive graphics display; among them are the light pen, the track ball, and the joy stick. The light pen is one of the most commonly used devices. It consists of a hand-held fiber-optic tube that can be "pointed" at a light source (e.g., cursor) on the screen and used to move the cursor to produce a line. The joy stick and track ball are both analog devices. The joy stick resembles a pilot's control stick. The track ball consists of a recess-mounted sphere that can be rotated by the palm of the hand. Both are equipped with potentiometers, whose outputs are converted to digitized x- and y-values, so that the cursor on the screen moves in direct proportion to the movement of the device.

The Phase II project was undertaken to evaluate the light pen, track ball, and joy stick for types of drawing tasks representative of Sketch Model applications. Accordingly, the primary experimental null hypothesis was that there is no significant difference among devices. The experimental procedure consisted of having the subjects use each of the three control devices to draw as perfect a circle and as perfect an equilateral triangle as possible.

A criterion function combining the relative importance of curved (circle test) versus straight line (triangle test) performance was devised. A parametric sensitivity analysis was carried out to investigate the dependency of the superiority of each control device on the relative importance of curved versus straight lines. It was concluded that the track ball produced superior "drawing" performances for all reasonable weights of straight versus curved lines. Thus, the track ball was selected as the best control device for implementing the Sketch Model procedure and was used in all later research.

In the Phase I research, operators were shown sets of points randomly sampled from known bivariate Gaussian probability density functions and asked to sketch (on the display) their estimates of the iso-probability contours of the parent distribution. The study indicated that humans can produce accurate (i.e., competitive with maximum likelihood estimation) Sketch Models of well-behaved continuous functions.

In the real world, however, functional relationships, though they may be "continuous in at least one dimension," are seldom "well-behaved," nor can they always be fully specified analytically. The Phase III research was undertaken to extend the Sketch Model concept in two directions. First, the Sketch Model was applied to a tactical problem in which the function to be estimated was not well behaved. The function, instead, was not only multidimensional; it was also multimodal and unsymmetric. Second, the usefulness of decisions reached with the aid of Sketch Models was investigated.

The problem used to study the usefulness of the Sketch Model as a decision aid was that of selecting the best flight path for an air strike against an island, given (1) known locations and suspected types of enemy sensors, and (2) predetermined aircraft fuel allotments and speed versus fuel consumption characteristics. Enemy sensor performance was modeled in terms of detection rate as a function of distance from the sensor. A Sketch Model consisted of a set of the iso-detection rate contours of the composite detection rate surface produced by four sensors at given locations, but whose detection rate versus range capability may be imperfectly known.

The goodness of a strike path was measured by a utility function that reflected a trade-off between minimizing the probability of being detected along the path and maximizing the fuel remaining at the target. The primary functions for specifying the best strike path were modeling the composite detection rate surface produced by the enemy's sensors and optimizing the strike path with respect to the model. Four system concepts were defined, representing different allocations of these two functions:

(1) modeling (without Sketch Model procedure) and optimization both allocated to the operator; (2) modeling (via the Sketch Model procedure) and optimization (aided by the Sketch Model) to the operator; (3) modeling (via Sketch Model procedure) to the operator and optimization (by a grid-oriented dynamic programming routine) to the computer; and (4) both modeling and optimization to the computer. This last allocation scheme provided what amounts to "answers" to the problem set.

Analyses of variance were performed on strike path utility data to compare system concepts. The results were weakened by the small number of subjects (4) available to provide the data. The conclusions evaluating the tactical usefulness of the decision aid were further weakened due to the problem set being too easy for the subjects. Within the qualifications posed by the small number of subjects and easy problem set, the result of the investigation of Sketch Model usefulness was that strike paths produced by computerized optimization operating on Sketch Models were significantly better than strike paths specified by subjects without the aid of Sketch Models.

The investigation of the accuracy with which humans could produce Sketch Models of "messy functions" was not affected by the easiness of the problem set. Sketch Model error was measured in terms of percent volume error from the true detection surfaces. Based on an examination of factors contributing to Sketch Model error, the findings relating to Sketch Model accuracy are:

1. Humans can use the present Sketch Model method to develop accurate models of multimodal, unsymmetric, three-dimensional functions.

2. Considerable improvement over present levels of accuracy can be achieved since the major sources of error can be reduced by refining the methodology.

3. Humans can use the Sketch Model method to significantly reduce the effects of uncertain information.

Phase IV research is presently in progress and will investigate the ability of automatic optimization or operator aided optimization procedures to use Sketch Models within their criterion functions. The Phase IV research will also measure the performance improvement resulting from allowing operators to guide, constrain, and control computer based optimization procedures.